Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Analýza sentimentu na sociální síti Twitter

Plzeň 2019 Jakub Hain

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 25. dubna 2019

Jakub Hain

Abstract

This thesis focuses on sentiment analysis and data summarization. Final use of the application should be for social network Twitter in Czech language.

The text shows several machine learning methods used for classification (Naive Bayes, Support Vector Machine and Maximum Entropy). Extractive summarization is implemented as Latent semantic analysis. Selection of sentences is implemented in 2 versions (Gong and Liu, Steinberger and Ježek).

Sentiment was tested on classified data from Czech Facebook and English tweets. Testing on Czech data shows comparable results on all classifiers except SVM.

Summarization was tested on English dataset. ROUGE-1 metric was used for verification. Both versions of LSA performed similarly.

Tato práce je zaměřena na analýzu sentimentu a sumarizaci dat. Konečné využití aplikace by mělo být pro sociální síť Twitter v češtině.

V textu jsou popsány jednotlivé metody strojového učení použité pro klasifikaci (Naivní Bayes, Support Vector Machine a Maximum Entropy). Extraktivní sumarizace je implementována pomocí Latentní sémantické analýzy. Věty jsou vybrány dvěma způsoby (Gong a Liu, Steinberger a Ježek).

Sentiment byl testován na ohodnocených datech z českého Facebooku a anglických tweetech. Při testování na českých datech byly výsledky jednotlivých klasifikátorů srovnatelné kromě SVM.

Sumarizace byla otestována na anglickém datasetu. Pro ověření byla použita metrika ROUGE-1, podle které obě verze sumarizace dosahovaly podobných výsledků.

Obsah

1 Úvod

Cílem práce je vytvořit program, který bude schopný vyhodnotit sentiment příspěvků ze sociální sítě Twitter[1] a poté bude schopen vrátit sumarizovaný výstup. Úkolem práce bude prozkoumat formát příspěvku (tweet) na sociální síti Twitter. Dále bude potřeba zjistit možné způsoby sumarizace a prozkoumat jak jsou použitelné.

Aplikace bude schopna klasifikovat tweet do jedné ze 3 kategorií:

- pozitivní,
- neutrální,
- negativní.

Při průzkumu veřejného mínění je důležité také vědět, co je hlavními důvody pro pozitivní a negativní tweety. Toto by mělo být zjištěno sumarizací pozitivních a negativních tweetů. Neutrální tweety jsou ze sumarizace vynechány, protože nemají tak výrazný dopad při průzkumu názorů.

Ve druhé kapitole jsou popsány sociální sítě a jejich rozšíření v současném světě. Dále je zde popsána sociální síť Twitter a způsoby přístup k API Twitteru.

Třetí kapitola se zabývá jednotlivými metodami předzpracování a reprezentace textu. V této kapitole je popsán význam tokenizace, stemování atd. Zvoleným modelem pro reprezentaci textu je bag of words. Strojové učení a použité klasifikátory jsou vysvětleny ve čtvrté kapitole, tato kapitola je stěžejní pro pochopení hlavní funkcionality programu. Mezi zvolené klasifikátory patří:

- Naivní Bayes [8],

[1] Twitter: `www.twitter.com`

- Komplementární Naivní Bayes [17],
- Support Vector Machine [11],
- Maximum Entropy [14].

V šesté kapitole je popsána zvolená metoda sumarizace, která využívá Latentní sémantickou analýzu. Jsou implementovány dva různé způsoby volby výsledné sumarizace definované podle:

- Gong a Liu [6],
- Steinberger a Ježek [19].

Zbytek práce je poté věnován samotnému řešení aplikace. Do této části patří výběr knihoven a popis důležitých částí z implementace programu.

V kapitole výsledky jsou poté vyhodnoceny jednotlivé testovací scénáře. Pro sentiment bude testování provedeno na krátkých komentářích ze sociální sítě Facebook a na datech z anglického Twitteru.

2 Sociální sítě

Sociální sítě jsou v současné době jednou z nejnavštěvovanějším webových stránek. Sociální sítě jsou určeny pro komunikaci mezi lidmi a sdílení jejich názorů. V současné době se mezi nejvíce používané sociální sítě řadí Youtube, Facebook, Twitter, Instagram. Facebook je v současné době největší sociální sítí s 2.13 miliardami aktivních uživatelů viz tab. 2.1.

Tabulka 2.1: Počet aktiv. uživ./měsíc na sociálních sítích. Zdroje[1]

Sociální síť	**Aktiv. uživatelé/měsíc [milion]**
Facebook[2]	2130
Youtube[3]	1500
WhatsApp	1300
Instagram[4]	800
Tumblr	794
Twitter	330
Skype	300
LinkedIn[5]	260
Reddit	250

Vzhledem ke stále rostoucímu počtu aktivních uživatelů, jsou sociální sítě jedním z nejlepších zdrojů nefiltrovaných informací a názorů všeho druhu. Pokud by bylo možné tyto informace zpracovat, získali bychom užitečný přehled o určité problematice. Například při vydání nového produktu bychom mohli ze sociálních sítí získat ohlas na náš nový produkt.

[1] $https://bit.ly/2BIM30d$
$https://youtube.googleblog.com/2017/06/updates-from-vidcon-more-users-more.html$
$https://goo.gl/EtRAbw$
$https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-usersf$

[2] www.facebook.com

[3] www.youtube.com

[4] www.instagram.com

[5] www.linkedin.com

2.1 Twitter

Twitter byl založen v roce 2006 v USA. Zakladateli Twitteru jsou Jack Dorsey, Noah Glass, Biz Stone a Evan Williams. Twitter umožňuje uživateli komunikovat s okolím pomocí krátkých zpráv zvaných tweet. Tyto zprávy měly do 7. listopadu 2017 maximální délku 140 znaků. V dnešní době mají ve většině jazyků délku 280 znaků. Tweety mohou kromě textu obsahovat i obrázky, video a odkazy.

Twitter je se svými 330 miliony uživatelů jednou z největších sociálních sítí. Každý den uživatelé napíší v průměru 500 milionů tweetů.

Vzhledem k počtu uživatelů Twitteru a jejich aktivitě lze sledovat současné dění ve světě, nové trendy, názory na produkty atd.

2.1.1 Twitter API

Twitter API (Application Programming Interface) umožňuje přístup k datům z Twitteru. V současné době jsou 2 způsoby získání dat:

- Search API,
- Streaming API.

Search API umožňuje přístup k datům Twitteru a je založeno na REST (Representational state transfer) architektuře. Uživatelům je tedy umožněn přístup k jejich profilu, tweetům, seznamu přátel a odběratelů atd. Search API dovoluje uživateli vyhledávat tweety na základě dotazů. Dostupné jsou pouze příspěvky vytvořené za posledních 7 dní.

Streaming API je určeno pro získávání dat v reálném čase. Při použití tohoto způsobu je vytvořeno připojení na data stream, který lze filtrovat

pomocí klíčových slov. Twitter umožňuje přes streaming API získat 1% až 40% z celkového objemu zpráv Twitteru.

Pro získání přístupu k API je nutné zaregistrovat aplikaci [16] (vyžaduje přihlášení) na stránkách Twitteru. Po registraci jsou uživateli vydány klíče pro přístup k API. Kompletní dokumentace API je dostupná na [4].

2.2 Omezení přístupu k Twitteru

Při přístupu k API je nutné dodržovat limity určené společností Twitter. Po překročení limitu je spojení přerušeno. Tyto omezení jsou určeny pro zmírnění nároků na servery Twitteru a zamezují případnému pokusu o zneužití. Limity jsou děleny do 15 minutových úseků. Základní počet volání je 15 pokud není stanoveno jinak.

Twitter má 2 různé způsoby omezení počtu volání:

- Uživatel - uživatel má svůj vlastní limit,
- Aplikace - limit je určen celé aplikaci a všem jejím uživatelům.

Pro vyhledávání tweetů podle klíčových slov má uživatel omezení na 180 a aplikace 450 volání za 15 minut. Kompletní tabulku s maximálním počtem požadavků je možné nalézt v dokumentaci API Twitteru [4].

3 Předzpracování textu a jazykové modely

Před použitím klasifikátoru nebo sumarizace je nutné text předzpracovat. Mezi techniky předzpracování patří tokenizace, stemming, term frequency inverse document frequency. Tyto techniky se používají pro zlepšení přesnosti jazykových modelů. Data jsou poté reprezentována jako bag of words model.

V této kapitole je vysvětleno, jak uvedené metody předzpracování fungují. Jejich pochopení je důležité pro další části práce. Podrobnější popis zmíněných technik lze nalézt v [3].

3.1 Tokenizace

Prvním krokem předzpracování je rozdělení slov do tokenů. Token je jeden prvek, který je využívaný pro klasifikaci. Token může být slovo, URL adresa, emotikon atd. Tyto tokeny jsou poté základem bag of words modelu reprezentace dokumentu v práci je dokumentem uvažován každý tweet. Ukázka tokenizace:

- @Jakub 8. 4. 2018 bude hezky www.url.com #hezky

1. @Jakub
2. 8. 4. 2018
3. hezky
4. www.url.com
5. #hezky

3.2 Stemming

Stemming je běžně využívaným postupem ve strojovém učení při předzpracování textu. Slouží k odstranění koncovek slova a vrácení kmenu (stem). V každém jazyce mají slova několik různých tvarů. Tyto tvary jsou brány modelem jako různá slova a snižují přesnost daného modelu. Pro odstranění tohoto problému se využívá stemming. Cílem je sloučit různé tvary slova do jednoho kmene. Tím se zvýší frekvence daného slova a zároveň přesnost slovníku. Například:

- červený, červená -> červen,
- studentův, studentům -> student.

Jak je z příkladu vidět, nejedná se o základní tvar slova, ale o takovou část slova, která se při změnách tvaru nemění.

3.3 Bag of words

Bag of words je model reprezentující dokument jako vektor vlastností (feature vector) o statické délce. Tento model neuchovává žádné informace o pořadí slov nebo kontextu, v jakém se slova vyskytují. Pouze určuje, zda se slovo v daném dokumentu nachází a počet kolikrát se slovo v dokumentu vyskytlo. Například:

1. @Jakub bude hezké počasí #pěkně
2. @Jakub 8. 4. 2018 bude venku hezky #hezky

Nejprve je tokenizací vytvořen slovník unikátních slov. V našich větách se vyskytuje 9 unikátních slov, pokud nepoužijeme stemming a pouze 6 unikátních slov za použití stemmingu:

1. Jakub,
2. hezk,
3. počas,
4. pěkn,
5. 8. 4. 2018,
6. venk.

Samotný vektor je vytvořen vložením počtu, kolikrát se slovo vyskytuje ve větě na index příslušící danému slovu viz 3.1.

Tabulka 3.1: Ukázka reprezentace tweetu bag of words modelem

Dokum.\Slovo	**Jakub**	**hezk**	**počas**	**pěkn**	**8. 4. 2018**	**venk**
1.	1	1	1	1	0	0
2.	1	2	0	0	1	1

3.3.1 TF-IDF

Pro vylepšení modelu jsou počty slov nahrazeny jejich vahou [15]. Tyto váhy se snaží snížit vliv slov, která se vyskytují velmi často („ale" nebo „bude") a tedy nemají tak velký význam pro klasifikaci. Na rozdíl od toho slova („sentiment" nebo „klasifikace"), která se vyskytují málo mají větší význam. Je také pravděpodobné, že slova, která se nevyskytují často, budou souviset s určitým tématem. Popis vzorce pro výpočet TF-IDF (vzorec 3.1) viz níže.

$$tfidf_{t,d,D} = tf_{t,d} * idf_t \tag{3.1}$$

Term-frequency

První částí je term-frequency (tf), tato část určuje důležitost slova (term) v dokumentu. Existuje několik možností jak vypočítat $\mathrm{tf}_{t,d}$, kde t je slovo v dokumentu d. Možnosti reprezentace jsou následující:

1. Binární - 1 pokud se slovo vyskytuje jinak 0,
2. Počet výskytů slova v dokumentu $\mathrm{f}_{t,d}$,
3. Logaritmicky škálovaná frekvence (vzorec 3.2),

$$tf_{t,d} = \log(f_{t,d} + 1) \tag{3.2}$$

4. Počet výskytů slova dělený délkou dokumentu |d| (vzorec 3.3).

$$tf_{t,d} = \frac{f_{t,d}}{|d|} \tag{3.3}$$

V práci je využita možnost 4.

Inverse document frequency

Idf vyznačuje důležitost slova v kolekci dokumentů. Výpočet idf_t je popsán ve vzorci (vzorec 3.4), kde |D| je celkový počet dokumentů a d_t je počet dokumentů, ve kterých se vyskytuje slovo t.

$$idf_t = \log \frac{|D|}{d_t} \tag{3.4}$$

Příklad

Pro výpočet jsou použity věty z kapitoly 3.3. Každá věta je považována za samostatný dokument. Výpočet hodnot tf a idf je ukázán ve 3.5 na slovech

„Jakub" a „počas".

$$\begin{aligned} tf_{Jakub,1} = \frac{1}{4},\ tf_{Jakub,2} = \frac{1}{5} \\ tf_{počas,1} = 1/4,\ tf_{počas,2} = 0 \\ idf_{Jakub} = \log\frac{2}{2} = 0,\ idf_{počas} = \log\frac{2}{1} = 0,6931 \end{aligned} \tag{3.5}$$

Pro „Jakub" je idf 0, protože se vyskytuje ve všech dokumentech a tedy nemá pro kolekci žádný význam. Tf-idf pro toto slovo je vždy 0. Výpočet tf-idf je zobrazen v ukázce 3.6 na slově „počas".

$$\begin{aligned} tf - idf_{počas,1,D} = \frac{1}{4} * 0,6931 = 0,173275 \\ tf - idf_{počas,2,D} = 0 * 0.6931 = 0 \end{aligned} \tag{3.6}$$

3.4 N-Gram

N-gramy jsou použity pro upřesnění jazykového modelu. N-gramy pomáhají nalézt souvislosti mezi jednotlivými slovy. Pro klasifikaci textu bylo prokázáno, že n-gramy nízkého řádu jsou vhodnou vlastností.

N-gramy je možné vytvářet na celých slovech nebo jednotlivých znacích, jedná se o množinu s n počtu prvků.

Příklad pro n-gram, kde prvkem jsou slova:

Ukázka bigramu pro modelování jazyka.

Bigramy: Ukázka bigramu, bigramu pro, pro modelování, modelování jazyka

4 Strojové učení

V této kapitole je popsáno, co je strojové učení [18, 8, 11] a několik základních metod. Tyto metody jsou využity v aplikaci pro určení sentimentu. U každé metody je vysvětleno, jak funguje a důvod zvolení.

Strojové učení je jednou z disciplín umělé inteligence. Zabývá se algoritmy a technikami, které umožňují programu učit se nebo vykonávat věci, ke kterým nebyly přímo naprogramovány. Algoritmy strojového učení se rozdělují do 4 skupin:

1. Učení s učitelem (Supervised learning),
2. Učení bez učitele (Unsupervised learning),
3. Kombinace předchozích (Semisupervised learning),
4. Zpětnovazebné učení. (Reinforcement Learning)

4.1 Klasifikace

Klasifikace je jednou z úloh strojového učení. Tato úloha se zabývá rozdělením dat (texty, obrázky, signály apod.) do předem známých tříd (pozitivní, negativní...). Často používaným příkladem je třídění e-mailů tzv. spam filtr, dělí e-mail do dvou tříd (spam a ham).

Klasifikace využívá principů učení s učitelem. V případu klasifikace sentimentu je nutné pro vytvoření modelu mít dostupná data podle, kterých je model učen.

Klasifikátor rozděluje data do jednotlivých tříd podle natrénovaných vlastností. Tyto vlastnosti by měly zvoleny tak, aby co nejlépe reprezentovaly pozorované třídy. Na těchto vlastnost závisí přesnost celého modelu.

Z toho vyplývá, že klasifikátor pracuje v N rozměrném prostoru, kde N je počet vlastností.

Klasifikátor je tedy funkce, která přiřadí vzorku x třídu y (4.1).

$$y = f(x) \tag{4.1}$$

4.1.1 Sentiment

Klasifikace sentimentu je jednou z klasifikačních úloh. Cílem je vytvořit model, který je schopen klasifikovat vstupní data do minimálně dvou tříd (pozitivní, negativní). V práci jsou využity tři třídy. Jedná se tedy o úlohu, která rozlišuje polaritu dat na:

- Pozitivní,
- Negativní,
- Neutrální.

Klasifikaci sentimentu je možné využít například v Bussiness inteligence, kde může sloužit pro sbírání ohlasů na nový produkt nebo událost. Klasickým příkladem je klasifikace filmových recenzí.

4.2 Naivní Bayes

Naivní Bayesův klasifikátor [3] je pravděpodobnostní metoda. Bayesovské klasifikátory přiřazují nejpravděpodobnější třídu k danému vzorku popsaným vektorem vlastností. Vlastnostem jsou nezávisle přiřazeny třídy, do kterých vlastnost patří vzorec (4.2), kde C je třída a X je vektor vlastností (viz

oddíl 3.3). Nehledě na nerealistický předpoklad, že vlastnosti jsou nezávislé, je výsledný Naivní Bayesovský klasifikátor v praxi velmi úspěšný.

$$P(X|C) = \Pi_{i=1}^{n} P(X_i|C) \tag{4.2}$$

Naivní Bayesův klasifikátor je založen na Bayesovu teorému. Bayesova věta je jedna ze základních v teorii pravděpodobnosti. Za předpokladu dvou náhodných jevů X a C s pravděpodobnostmi P(X) a P(C), přičemž P(C) > 0, potom platí vzorec 4.3. Pro přiřazení dokumentu do některé z tříd, musí být nejprve spočítána pravděpodobnost pro každou třídu. Zkoumanému prvku je přiřazena třída s nejvyšší pravděpodobností (vzorec 4.4).

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \tag{4.3}$$

$$\hat{C} = \underset{C}{argmax} P(C|X) = \underset{C}{argmax}(P(X|C)P(C)) \tag{4.4}$$

Výhodou je jednoduchost, rychlost učení a klasifikace. Model je možné dále trénovat i během používání. Vzhledem k možnosti využití bez znalosti vlastností dat, je vhodný jako základní algoritmus pro prvotní analýzu. Často je využíván pro klasifikaci textů (detektor spamu, přiřazení tématu k článku).

Naivní Bayes byl zvolen pro svou jednoduchost a rychlost. Jak ukazují výsledky studií [13, 12], je naivní Bayes vhodný pro klasifikaci sentimentu.

4.2.1 Trénování naivního Bayese

Priorní pravděpodobnost třídy P(C) je vypočtena z počtu dokumentů a celkového počtu dokumentů náležící dané třídě. Zjišťujeme tedy, jaké procento dokumentů náleží do této třídy (vzorec 4.5). Nechť N_c je počet doku-

mentů třídy c a N_{dok} je celkový počet dokumentů v trénovací množině.

$$P(c) = \frac{N_c}{N_{dok}} \tag{4.5}$$

Pravděpodobnost jednotlivých slov P(w_i,c), kde w_i je slovo, kterému je počítána pravděpodobnost. Počet(w_i,c) je počet kolikrát se slovo vyskytlo v trénovací sadě. Slovník V je složen ze všech slov, i těch, které se nevyskytují ve třídě c. Vzorec pro výpočet je tedy následující:

$$P(w_i, c) = \frac{počet(w_i, c)}{\sum_{w \in V} počet(w, c)} \tag{4.6}$$

Pro zpřesnění modelu v situaci, kdy se slovo vyskytlo v 1 třídě, ale nevyskytlo se v ostatních, je pro zpřesnění modelu možné využít Laplaceovo vyhlazování. Například pokud se slovo „báječný" neobjevilo v pozitivních datech, ale vyskytlo se v negativních datech, bylo by slovo bráno jako negativní, i když ve většině případů se toto slovo vyskytuje v pozitivním kontextu. Po přidání Laplaceova vyhlazování získáme vzorec 4.7, kde k čitateli je přičtena 1 a ve jmenovateli je |V|, což je velikost slovníku.

$$P(w, c) = \frac{počet(w, c) + 1}{\sum(počet(w, c)) + |V|} \tag{4.7}$$

Pokud se slovo nevyskytlo v trénovacích datech vůbec, je dobré ho ignorovat a nepřiřazovat mu žádnou pravděpodobnost.

4.3 Komplementární Naivní Bayes

Jedná se o modifikaci Naivního Bayese podle článku [17]. V tomto článku autoři uvádí, že je možné pomocí několika úprav vytvořit algoritmus Naivního Bayese, který je schopen se vyrovnat s nevyvážeností tříd dat a tím zlepšit jeho přesnost.

Nechť $\vec{d}$ je vektor dokumentů (d_1, d_2, ..., d_j, ... d_n). Počet výskytů slova je poté vyjádřen, jako $d_{i,j}$, kde i je slovo v dokumentu j. A nechť $\vec{y}$ je vektor hodnocení dokumentu (y_1, y_2, ..., y_n). Hodnota $\delta_{i,k}$, kde i je slovo a k je dokument, se rovná 1 pokud se slovo vyskytuje v dokumentu, jinak rovna 0. Pak je výpočet Komplementárního Naivního Bayese následující:

1. $d_{i,j} = log(d_{i,j} + 1)$ // tf transformace
2. $d_{i,j} = d_{i,j} log(\frac{\sum_k 1}{\sum_k \delta_{i,k}})$ // idf transformace
3. $d_{i,j} = \frac{d_{i,j}}{\sqrt{\sum_k (d_{i,j})^2}}$ // normalizace
4. $\hat{\theta}_{c,i} = \frac{\sum_{j:y_j \neq c} (d_{i,j} + \alpha_i)}{\sum_{j:y_j \neq c} \sum_k (d_{k,j} + \alpha)}$ // komplement
5. $w_{c,i} = log(\hat{\theta}_{c,i})$
6. $w_{c,i} = \frac{w_{c,i}}{\sum_i w_{c,i}}$ // normalizace vah
7. Nechť t=(t_1, t_2, ...,tt_n) je testovací dokument a t_i je počet výskytů slova i
8. Dokument je ohodnocen podle: $l(t) = \underset{c}{argmin}\ (t_i w_{c,i})$

Tyto úpravy dle výsledků v článku [17] dosahují podobné úspěšnosti při klasifikaci, jako SVM na jimi použitých datech (20 newsgroup[1], Industry Sector a Reuters 21578[2]).

4.4 Maximální entropie

Klasifikátor maximální entropie [14] (zkráceně MaxEnt) je někdy nazývaný multinomiální logistická regrese. MaxEnt patří do rodiny klasifikátorů

[1] `http://qwone.com/~jason/20Newsgroups/`
[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

známých jako exponenciální nebo log-lineární klasifikátory. Na rozdíl od Bayese se jedná o diskriminativní model. Klasifikátor je založen na principu maximální entropie, tedy ze všech modelů, které odpovídají trénovacím datům zvolí ten, který má největší entropii (vzorec 4.8).

Pro vytvoření modelu je nutné mít anotovaná data, jedná se tedy o metodu učení s učitelem. Ze vstupních dat se vytvoří množina vlastností. Každé vlastnosti je přiřazena váha, se kterou patří do určité třídy c. Při výběru vhodné třídy je tedy snaha maximalizovat pravděpodobnost třídy c za výskytu vlastnosti x.

$$\hat{c} = \underset{c}{argmax} P(c|x) \tag{4.8}$$

Pro výpočet se používá vzorec 4.9, kde w_i je váha slova a f_i je vlastnost. Z je normalizační hodnota, která omezuje výpočet na rozmezí <0,1>.

$$P(c|x) = \frac{1}{Z} \exp(\sum_i w_i f_i(c, x)) \tag{4.9}$$

Po dosazení za normalizační faktor Z a specifikaci počtu vlastností N je konečný vzorec pro výpočet pravděpodobnosti 4.10.

$$P(c|x) = \frac{\exp(\sum_{i=1}^{N} w_i f_i(c, x))}{\sum_{c' \in C}(\exp(\sum_{i=1}^{N} w_i f_i(c', x)))} \tag{4.10}$$

Pro nalezení optimálních vah je možné využít několika metoda např. stochastický vzestup gradientu, L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno). Tyto metody začínají s nulovými vahami a posouvají se ve směru gradientu.

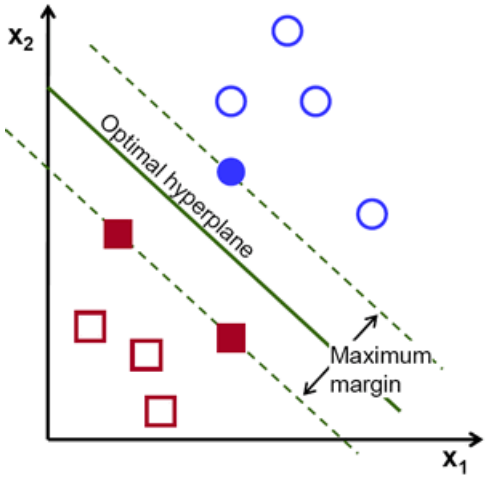
Algoritmus byl zvolen, jako zástupce logistické regrese.

4.5 Support vector machine

Support vector machine (SVM) [20, 11] je metoda učení s učitelem. Stejně jako MaxEnt se jedná o diskrimanativní algoritmus, tedy negeneruje se model

jak data vznikla, ale rozhodující hranice podle trénovacích dat. Může být použita pro klasifikační a regresní problémy. Vstupní data jsou zobrazena jako bod v n-dimenzionálním prostoru, kde n je počet vlastností (např. velikost slovníku). Každý tento bod má přiřazenou jemu odpovídající hodnotu.

Při trénování je hledána nadrovina (obr. 4.1), která odděluje třídy od sebe. Nadrovina by měla co nejlépe oddělovat třídy od sebe. V základu je



Obrázek 4.1: Optimální nadrovina Zdroj[3]

SVM (Support Vector Machine) lineární učící systém, pro klasifikování do dvou tříd. Nechť sadou trénovacích vzorků D je

$\{(x_1, y_1),(x_2, y_2), ..., (x_n, y_n)\}$.

Kde $x_i = (x_{i1}, x_{i2}, ...,x_{ir})$ je r-dimenzionální vstupní vektor a $y_i \in 1, -1$ je třída, do které vzorek x_i patří.

Pro sestavení klasifikátoru hledá lineární SVM lineární funkci ve tvaru 4.11, kde w je vektor vah a b je bias (předpojatost).

$$f(x) = \langle \vec{w} \cdot \vec{x} \rangle + b \tag{4.11}$$

[3] `https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html`

Takže vstupnímu vektoru x_i je přiřazena pozitivní hodnota, pokud f(x_i)$\geq$ 0 a negativní, pokud f(x_i)$\lneq$ 0. $\langle w \cdot x \rangle$ (vzorec 4.12) je skalární součin vektorů w a x.

$$y_i = \begin{cases} 1 & pokud \langle \vec{w} \cdot \vec{x} \rangle + b \geq 0 \\ -1 & pokud \langle \vec{w} \cdot \vec{x_i} \rangle + b \lneq 0 \end{cases} \tag{4.12}$$

SVM tedy hledá nadrovinu definovanou vzorcem:

$$\langle w \cdot x \rangle + b = 0 \tag{4.13}$$

Z geometrického hlediska rozděluje nadrovina prostor dat na dvě části, jednu část pro pozitivní data a druhou pro negativní.

Data často nejsou lineárně separovatelná a proto se používají nelineární kernely (funkce jádra) K pro určení nadroviny. Existuje několik možných nadrovin (obr. 4.2):

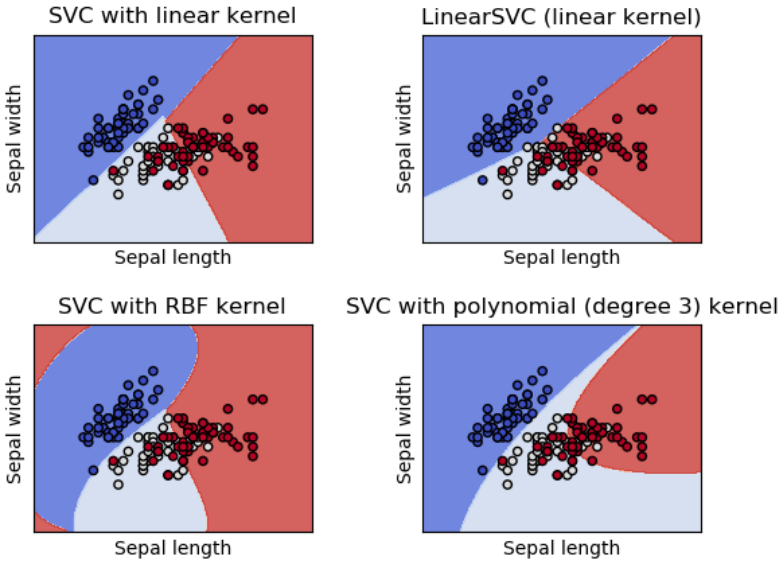
- polynomiální,
- radiální bázová funkce,
- hyperbolický tangent.

Pro klasifikaci do více tříd je možné využít jednu z následujících možností:

- One vs All,
- One vs One.

Často používaná je metoda one vs all, která vytvoří n klasifikátorů, kde n je počet tříd. Pro každou třídu c je tedy vytvořen klasifikátor, kde pozitivní data náleží třídě c a negativní jsou všechny ostatní třídy.

[4] $http://scikit-learn.org/stable/auto_examples/svm/plot_iris.html$



Obrázek 4.2: Ukázka nadrovin při různých volbách jádra. Zdroj [4]

One vs one vytváří více klasifikátorů (vzorec 4.14, kde |C| je počet tříd) něž one vs all, ale vzhledem k menší velikost trénovacích dat jednotlivých tříd může být jeho trénování rychlejší.

$$\frac{|C|(|C|-1)}{2} \tag{4.14}$$

Důvody pro výběr tohoto algoritmu jsou následující. SVM je vhodné pro data s vysokou dimenzionalitou (vysoký počet vlastností). Support vector machine bylo v mnoha studiích [14, 13] úspěšně použito pro klasifikaci sentimentu.

5 Sumarizace textu

V této kapitole je popsán algoritmus pro sumarizaci textu. Sumarizace je použita pro generování reprezentativního výstupu pro pozitivní a negativní tweety.

Popsány budou dva způsoby sumarizace:

1. Latentní sémantická analýza,
2. Latentní Dirichletova analýza.

V práci je pak implementována pouze 1. metoda.

Existují dva hlavní typy pro automatickou sumarizaci textu extraktivní a abstraktivní. Extrakt je souhrn tvořený frázemi, větami nebo odstavci z původního dokumentu. Extrakty trpí chabou souvislostí zařazených úseků. Výběr vět může být proveden bez ohledu na kontext, výsledek bývá nevyvážený a nesourodý.

Abstrakt je souhrn, který nemusí nutně obsahovat sekvence slov z originálního dokumentu. V současné době stále těžko řešitelná úloha.

5.1 Latentní sémantická analýza

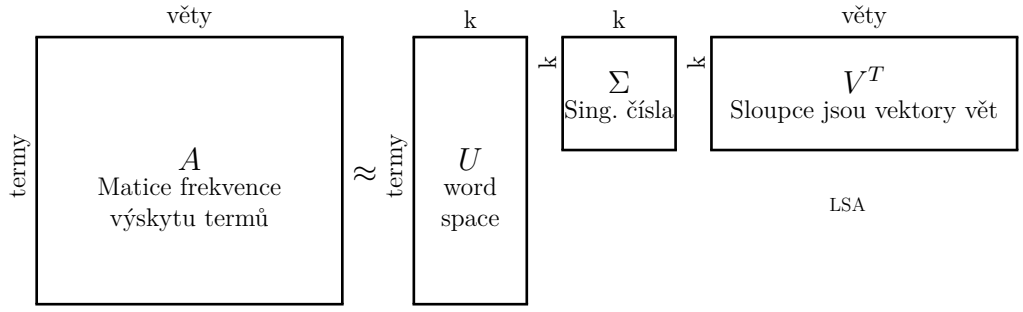
V roce 2001 publikovali Yihong Gong a Xin Liu svou myšlenku na využití latentní sémantické analýzy (LSA) pro sumarizaci textu [6]. Inspirování latentním sémantickým indexováním, použili singulární rozklad, anglicky singluar value decomposition (SVD) na sumarizaci textu.

Proces začíná vytvořením m $\times$ n matice A[A_1, A_2, ..., A_n], kde m je počet termů a n je počet vět. Každý sloupec A_i je vektor věty i v dokumentu obsahující hodnoty frekvence výskytu termu.

Pokud je matice A m × n, kde bez ztráty generality m≥n , pak je singulární rozklad A definován jako:

$$A = U\Sigma V^T, \tag{5.1}$$

kde U = [u_{ij}] je m × n sloupcově ortonormální matice jejíž sloupce jsou zvány levý singulární vektor. Σ = diag(σ_1, σ_2, . . . , σ_n) je diagonální matice o rozměru n × n. Prvky na diagonále jsou pozitivní singulární hodnoty seřazené v sestupném pořadí. V = [v_{ij}] je n × n ortonormální matice, jejíž sloupce se nazývají pravé singulární vektory. Rozměr matic je redukován na k dimenzí, kde k < n, takže U je redukována na m × k, Σ na k × k a V^T na k × n ()obr. 5.1).



Obrázek 5.1: Ukázka singulárního rozkladu

Po provedení SVD na matici A, je v redukované matici V mapováno k nejvýznamnějších témat.

5.1.1 Gong a Liu

Tato metoda je založena na nalezení nejvýznamnější reprezentace pro každé téma. V praxi to znamená, že v každé řádce matice V^T hledáme index s nejvyšší hodnotou.

5.1.2 Steinberger a Ježek

Vylepšení předchozí metody. Gong a Liu metoda zvolí pouze věty, které vhodně reprezentují 1 téma. Takto zvolená věta, ale nemusí být pro sumarizaci vhodná. Steinberger a Ježek vytvořili modifikaci volby věty [19], která bere v úvahu i další témata. Věty jsou ohodnoceny podle následujícího vzorce:

$$s_k = \sqrt{\sum_{i=1}^{n} v_{k,i}^2 \cdot \sigma_i^2}, \tag{5.2}$$

kde s_k je délka vektoru k-té věty, n je počet dimenzí nového prostoru. Tato hodnota je nezávislá na počtu zvolených vět. Steinberger a Ježek navrhují volbu počtu dimenzí n jako počet hodnot vyšších než polovina nejvyšší hodnoty σ.

Do výsledného souhrnu jsou pak zařazeny věty jejichž hodnota s_k je nejvyšší. Výstupem sumarizace v aplikaci je 5 tweetů pro pozitivní a 5 tweetů pro negativní sentiment.

6 Výběr prostředků

V této kapitole budou popsány použité knihovny, vstupní a výstupní formáty a celkový návrh aplikace.

6.1 Java

Pro řešení problému byl zvolen jazyk Java vyvíjený společností Oracle. Při výběru jazyka bylo uvažováno o Pythonu[1] a Javě. Python je vhodnější pro strojové učení a matematické úlohy. Pro jazyk Java i Python existují knihovny, které implementují potřebné algoritmy. Pro vytvoření programu byl nakonec zvolen jazyk Java ve verzi 1.8[2]. Především kvůli lepší znalosti Javy v době volby programovacího jazyka.

6.2 Twitter API

Při výběru způsobu pro přístup k API Twitteru byly uvažovány 2 knihovny JTwitter[3] a Twitter4J[4]. Obě knihovny umožňují přístup k search i streaming API. Pro Twitter4J bylo rozhodnuto převážně kvůli lepší dokumentaci a rozsáhlému zdroji informací.

Tabulka 6.1: Použité parametry pro vyhledávání tweetů

Parametr	Požadováno	Popis
q	ano	Vyhledávaný pojem, max. délka 500 znaků
geocode	ne	Zeměpisná délka a šířka a vzdálenost pro omezení oblasti tweetů
lang	ne	Jazyk tweetu
since_id	ne	Vrací pouze tweety s id větší než zadané

6.2.1 Vyhledávání příspěvků

Vyhledávání pomocí search API je umožněno voláním GET na endpoint `https://API.twitter.com/1.1/search/tweets.JSON`. Toto volání může mít několik parametrů, parametry použité v aplikaci jsou uvedeny v tab. 6.1. Parametr since_id je kvůli limitům API, pro pokračování stahování od určitého id. Zbytek parametrů je možné najít v dokumentaci API [4]

6.2.2 Formát příspěvku

Po vyhledávání je vrácen JSON soubor obsahující jednotlivé tweety (ukázka 6.1). Tyto tweety obsahují informace o uživateli, odkazech, zda se jedná o retweet atd.

Ukázka 6.1: Zkrácená ukázka výsledku vyhledávání

```
{"statuses":[
   {
      "created_at": "...",
      "id": "...",
      "text": "...",
      "entities":{
```

[1] `https://www.python.org/`

[2] `https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`

[3] `https://www.winterwell.com/software/jtwitter.php`

[4] `http://twitter4j.org/en/index.html`

```
        "hashtags": [],
        "symbols": [],
        "user_mentions": [],
        "urls": []
      },
      "user"{...}
      ...
   }
],
"search_metadata"{..}
}
```

Popis některých položek:

- statuses: pole jednotlivých výsledků vyhledávání,
- status: položka obsahující veškeré informace o tweetu (id, text, lokace, retweet, počet oblíbení, populární atd.)
- user: obsahuje informace o uživateli(jméno, lokace, popis, počet přátel, verified atd.),
- seach_metadata: informace o parametrech vyhledávání (od id, max id, další výsledky).

Z těchto informací je pro sentiment a sumarizaci využit pouze text tweetu.

6.2.3 Twitter4J

Twitter4J je jednou z nejpoužívanějších knihoven pro přístup k Twitter API v javě. Tato knihovna je distribuována pod licencí Apache 2.0. Umožňuje plné využití API Twitteru. Podporuje OAuth authentifikaci. Knihovna je

dostupná na webové stránce vývojáře[5], stejně jako dokumentace a ukázky použití.

6.3 Tokenizace

Pro tokenizaci byla volba mezi Apache Lucene[6] a Twokenizer[7]. Twokenizer je vytvořen přímo pro Twitter a je schopný rozlišit mezi jednotlivými částmi příspěvku na Twitteru. Apache Lucene je více flexibilní knihovnou, kterou je možné využít i pro další kroky předzpracování. Konečné rozhodnutí vyplynulo ve prospěch knihovny Twokenizer.

Upravený twokenizer je v aplikaci využit pro rozpoznání jednotlivých typů prvků vyskytujících se ve tweetu (url, emotikon, mention, slovo, tag). V twokenizeru bylo nutné upravit rozpoznávání emotikonů, které v původní verzi vykazovalo celkem podstatné chyby.

6.4 Stemming

Pro stemmning byla pro češtinu volba mezi Apache Lucene a HPS: High Precision stemmer, dále jen HPS, vytvořený na Západočeské Univerzitě, tento stemmer je detailně popsán v článku [1]. U HPS je výhodou jeho vytvoření pro češtinu, lepší možnost nastavení a pokročilejší algoritmy pro stemming. Pro Lucene hovoří snazší použitelnost pro různé jazyky, na rozdíl od HPS, kde je nutné mít pro každý jazyk vytvořený jazykový model. Proto byl ve výsledném hodnocení zvolen Apache Lucene.

[5] `http://twitter4j.org/en/index.html`

[6] `http://lucene.apache.org/`

[7] `https://github.com/vinhkhuc/Twitter-Tokenizer/blob/master/src/Twokenizer.java`

6.5 Knihovna pro SVD

Volba knihovny pro SVD byla problematická. Bohužel v době psaní práce neexistuje knihovna, která by umožňovala výpočet SVD za použití řídké matice. Toto je vzhledem k omezené paměti Javy problém. Hlavní rozhodování probíhalo mezi EJML[8] (Efficient Java Matrix Library) a knihovnou Colt[9] od společnosti Cern. Knihovna EJML vychází z dostupných benchmarků[10] jako rychlejší[11] a efektivnější[12].Proto byla zvolena knihovna EJML.

6.6 Uživatelské rozhraní

Grafické rozhraní k aplikaci je vytvořeno v JavaFX[13]. V JavaFX existují dva způsoby tvorby GUI:

- FXML soubor pro definování elementů GUI a jejich interakce,
- vytváření elementů v kódu.

Pro vytvoření aplikace byla zvolena možnost druhá Převážně z důvodů předchozích zkušeností s tímto způsobem vytváření a její jednoduchosti.

Rozhodování při tvorbě GUI bylo mezi Java Swing a JavaFX. Java Swing je starší technologie než JavaFX. Pro Swing je dostupné více komponent a mnoho knihoven, které přidávají další komponenty a rozšiřují funkcionalitu. JavaFX je modernější technologií, umožňuje snazší vytváření různých modifikací vzhledu pomocí CSS (Cascade Style Sheets) a jedná se o nástupce

[8] `http://ejml.org/wiki/index.php?title=Main_Page`
[9] `http://dst.lbl.gov/ACSSoftware/colt/`
[10] `http://lessthanoptimal.github.io/Java-Matrix-Benchmark/`
[11] `https://github.com/optimatika/ojAlgo/wiki/SVD`
[12] `http://lessthanoptimal.github.io/Java-Matrix-Benchmark/manual/DescriptionMemory/`
[13] https://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html

Swingu. Vzhledem k novosti technologie není nabídka komponent tak široká, ale jsou dostupné všechny komponenty potřebné pro vytvoření GUI pro tuto aplikaci.

7 Vstupní a výstupní soubory

Tato kapitola popisuje formáty vstupních a výstupních souborů včetně ukázek.

7.1 Formát vstupních souborů

Aplikace je schopna zpracovat několik různých typů vstupu. Formáty souborů, které je aplikace schopna přečíst jsou: XML, TXT, CSV a JSON. Sentiment musí být označen jedním ze 3 způsobů, každý formát požaduje určitý typ (tab. 7.1):

1. positive, neutral, negative,
2. p, 0, n,
3. 1, 0, -1.

Tabulka 7.1: Typy reprezentace sentimentu pro různé formáty, JSON je generován přímo z objektu a předpokládá jiný předpis

Typ / **Formát**	1.	2.	3.
XML	ne	ne	ano
TXT	ano	ano	ano
CSV	ano	ano	ano

Pro XML (viz ukázka 7.1) musí být přítomen základní tag `<tweets>`, který obklopuje všechny tweety. Každý tweet je poté ohraničen tagem `<tweet>`, který má jako atribut polaritu a jeho vnitřkem je tag text obsahující text tweetu.

Ukázka 7.1: Ukázka trénovacího/testovacího XML souboru

```
<tweets>
   <tweet polarity=1>
      <text>Ukazka textu tweetu @twitter</text>
   </tweet>
</tweets>
```

TXT umožňuje několik způsobů reprezentace. V tomto souboru jsou jednotlivé záznamy odděleny novou řádkou.

1. 1 soubor se všemi tweety a jejich polaritou
2. 2 soubory: soubor s polaritou tweetů a soubor s textem

Pro 1. způsob (viz ukázka 7.2) je polarita prvním slovem v souboru. Pokud první slovo neodpovídá jednomu z typů zápisu sentimentu jedná se o neplatný formát a soubor není zpracován.

Ukázka 7.2: Ukázka 1. typu TXT souboru

```
positive Lorem ipsum dolor et sit amet
```

Pro druhý způsob jsou potřeba dva soubory, kde první soubor obsahuje texty a druhý soubor obsahuje polaritu.

CSV je velmi podobné TXT souboru, polarita je v tomto formátu oddělena od textu tabulátorem. Pro případ, že by se v textu vyskytoval další tabulátor, jsou veškeré další tabulátory v řádce ignorovány.

JSON je generován přímo z objektů Javy. Pro načtení tohoto formátu je použita knihovna GSON[1], tuto knihovnu vytvořila společnost Google. inc. Umožňuje načítání a ukládání Java objektů z nebo do formátu JSON. Při volbě vhodné reprezentace byla možnost volby mezi serialized objektem a vytvořením JSON souboru (viz ukázka 7.3). Nakonec byl zvolen formát JSON, protože ho lze upravit a je tedy více variabilní při používání.

[1] `https://github.com/google/gson`

Ukázka 7.3: Ukázka trénovacích/testovacích dat v jsonu

```
[{
   "sentiment": "NEGATIVE",
   "tweet": {
      "text": "Lorem ipsum dolor sit amet"
   }
},
{
   "sentiment": "POSITIVE",
   "tweet": {
      "text": "Excepteur sint occaecat cupidatat non proident"
   }
}]
```

7.2 Formát výstupních souborů

Jako výstup aplikace jsou brány ohodnocené tweety a sumarizace ohodnocených tweetů. Každou z těchto položek je možné uložit do souboru.

7.2.1 Výstup klasifikace sentimentu

Pro sentiment je výstupním formátem JSON (viz ukázka 7.4). Formát byl zvolen pro jeho jednoduchou možnost editace a podporu z pohledu programovacích jazyků. Pro vytvoření tohoto výstupu je využita knihovna GSON, která umožňuje převedení java objektu do formátu JSON. Položka user je plně volitelná a nemá pro běh aplikace a klasifikaci velký význam. Povinná část je tedy pouze text. Sentiment je v tomto souboru také nepovinnou součástí, je vyplněn pouze pokud byl tweet klasifikován.

Ukázka 7.4: Ukázka výstupu klasifikovaných dat v jsonu

```
[{
```

```
"sentiment": "NEGATIVE",
"user":{
   "name": "Anonym",
},
"text": "Lorem ipsum dolor sit amet"
},
{
"sentiment": "POSITIVE",
"user":{
   "name": "Anon",
},
"text": "Excepteur sint occaecat cupidatat non proident"
}]
```

7.2.2 Sumarizace

Výsledek sumarizace je stejně jako sentiment ukládán do souboru ve formátu JSON (viz ukázka 7.5). Obsahuje veškeré informace o použité metodě sumarizace a případně její verzi. Sumarizace je poté rozdělena na negativní a pozitivní sumarizaci.

Ukázka 7.5: Ukázka výstupu sumarizace dat v jsonu

```
{
"method": "LSA",
"version": "Steinberger&Jezek",
"positive": ["Excepteur sint occaecat cupidatat non proident"]
"negative": ["Lorem ipsum dolor sit amet"]
}
```

8 Řešení

V této kapitole je popsáno samotné řešení aplikace s ukázkami kódu a odůvodnění jednotlivých rozhodnutí. Jsou zde vyzdviženy důležité části aplikace a některé zajímavé poznatky.

Kompletní uml diagram je možné nalézt na přiloženém DVD, nebo v příloze B.

8.1 Konfigurace programu

Jednou ze stěžejních částí při vytváření programu byla možnost konfigurace jednotlivých částí. Z tohoto důvodu byl vytvořen konfigurační soubor s názvem `settings.ini`, pro vytvoření tohoto souboru byla použita knihovna Ini4J[1]. Ukázkový konfigurační soubor lze nalézt v příloze D.

- lang
  - umožňuje nastavit jazyk pro stemming
  - cz - čestina
  - en - angličtina
- font_size
  - určuje velikost textu v GUI
- colors
  - umožňuje nastavit barvy textu ve sloupci s textem tweetu
  - jednotlivé položky: tag ,mention, word, url, emote

[1]Odkaz: http://ini4j.sourceforge.net/

- type
  - určuje typ použitého klasifikátoru (NB = 0, MAXENT = 1, SVM = 2, NBTFIDF = 3, CTFIDFNB = 4)
- sumarizace
  - method - použitá metoda sumarizace
  - version - verze LSA (Gong&Liu = 0, Steinberger&Ježek = 1)
- svm
  - type - typ svm (c-svm = 0, nu-svm = 1)
  - kernel - typ jádra svm (lineární = 0, polynomiální = 1, radiální báze = 2)
  - nu - hyperparametr svm <0,1>, využit pokud je nastaven typ na nu-svm
  - c - bezrozměrný hyperparametr svm, pro c-svm

8.2 Třída Tweet

V této třídě budou uloženy jednotlivé tweety a související informace. Na této třídě je postavena celá aplikace a proto je nutné znát její základní strukturu (ukázka 8.1) pro další práci s aplikací a pochopení k čemu slouží.

Ukázka 8.1: Proměnné třídy Tweet

```
/*informace o useru*/
private User user;
/*text tweetu*/
private String text;
/*slova pro zobrazeni v GUI podle typu*/
private List<Token> tokens;
```

8.3 Načtení dat z Twitteru

Data jsou ze sociální sítě Twitter načítána v souladu s podmínkami použití API. Aplikace je schopna vyhledat data podle zadaného dotazu např. „Západočeská univerzita" vrátí všechny tweety obsahující tento termín za poslední týden. Vyhledávání neumožňuje vyhledat retweety z důvodu duplikace hodnot.

Existují dva způsoby získání dat, první způsob umožňuje zvolit počet tweetů, které chceme stáhnout. Pokud tento počet není dosažen jsou vráceny všechny získané tweety. Druhý způsob vrátí prvních 500 získaných tweetů nebo méně, pokud není dosaženo 500.

Implementace těchto metod je ve třídě `TweetDownloader`, která implementuje rozhraní `ITweetDownloader`(ukázka 8.2).

Ukázka 8.2: Rozhraní ITweetDownloader

```
List<Tweet> searchTweetsCount(String query, int count);
List<Tweet> searchTweets(String query);
```

Dotaz je tedy vytvářen ve tvaru (ukázka 8.3):

- `Dotaz -filter:retweets -filter:nativeretweets`,

Filtr retweets odstraňuje retweety vytvořené uživateli, kteří zmáčkli tlačítko retweet. Nativeretweets vyfiltruje klasické (manuální) retweety.

Ukázka 8.3: Kostra metody searchTweets()

```
public List<Tweet> searchTweets(SearchQuery query) {

   List<Tweet> tweets = new ArrayList<>();
   Query q = new Query(query.getSearchQuery().toLowerCase() + "
       -filter:retweets -filter:nativeretweets");
```

```
    q.count(10);
    if(query.getLatitude() != null &&query.getLongitude() != null) {
    q.setGeoCode(new GeoLocation(query.getLatitude(),
        query.getLongitude()), query.getRange(), Query.Unit.km);
    }

    if(query.getCount() != 0){
       this.requiredTweets = query.getCount();
    }
    if(query.getLanguage() == Language.Czech) {
       q.setLang("cs");
    }else if(query.getLanguage() == Language.English){
       q.setLocale(("en-EN"));
    }
    search(tweets,q);
    this.requiredTweets = 500;
    return tweets;
}
```

8.4 Předzpracování textu

Předzpracování textu je rozděleno do několika kroků:

1. rozdělení textu do tokenů,
2. odstranění písmen, které se vyskytují více než dvakrát po sobě (góóóóól => góól),
3. stemming slova,
4. vytvoření n-gramů.

8.4.1 Stemming

Stemmer je zvolen podle jazyka v nastavení aplikace, pokud by jazyk nebyl nastaven nebo by aplikace jazyk nerozeznala, byl by použit výchozí stemmer `StandardAnalyzer`, který tvar slova nijak nemění. `StandardAnalyzer` je v aplikaci využit při předzpracování textu pro zobrazení v GUI.

Stemming je použit pouze na normální slova (není použit na url, tag a mention). Při použití stemmingu na ostatní typy dochází k jejich změnám a odstranění jim specifických částí.

V ukázce 8.4 je zobrazena jediná část programu, která je závislá na jazyku, pokud by byla aplikace v současné podobě rozšířena o podporu dalších jazyků, byly by jednotlivé implementace stemmeru doplněny na toto místo v kódu.

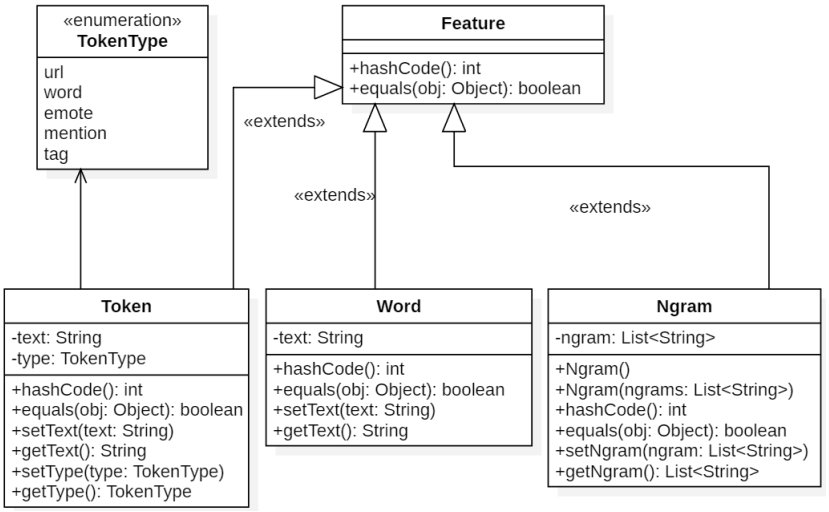
Ukázka 8.4: Volba stemmeru podle zvoleného jazyka

```
Analyzer analyzer = new StandardAnalyzer();
if(language == Language.English){
   analyzer = new EnglishAnalyzer();
}else if (language == Language.Czech){
   analyzer = new CzechAnalyzer();
}
```

Stemmeru je možné nastavit seznam stopslov. Stopslova jsou slova, které v textu nejsou významová, mají obvykle především syntaktický význam např. spojky a předložky. Při předzpracování textu pro klasifikaci je použit výchozí seznam poskytnutý knihovnou Lucene.

8.4.2 Vytváření vlastností

Metody pro vytvoření vlastností se nacházejí ve třídě `CreateFeature`. Tato třída má umožňuje dva způsoby volání:



Obrázek 8.1: UML diagram pro vlastnosti

```
List<Feature> createFeatures(List<Token> token)
List<Feature> createFeatures(Tweet t)
```

Pro vytvoření n-gramů je použita třída `Ngrams`, která je schopna vytvořit n-gramy o libovolné délce, v programu je využita délka dva a tři. Tato třída má následující metody:

```
List<Feature> createNgrams(List<Token> tokens, int length)
List<Feature> createNgrams(Tweet tweet, int length)
```

Pro uložení n-gramů je vytvořena třída `Ngram`. Unigramy jsou uloženy do instancí třídy `Word`, i když by bylo možné vše ukládat jako n-gram, není toto řešení vhodné, kvůli větší velikosti objektu v paměti a jeho vyšší složitosti (string vs list). Obě tyto třídy dědí od třídy `Feature` (viz obr. 8.1)

8.5 Klasifikátory sentimentu

Klasifikátory implementují rozhraní `IClassifier`, které zajišťuje přesnou strukturu jednotlivých tříd. Každý klasifikátor využívá následující vlastnosti:

- unigram,
- bigram,
- trigram,

Pro přidání nového klasifikátoru je tedy vždy implementováno rozhraní `IClassifier`, jehož metody jsou zobrazeny v ukázce 8.5

Ukázka 8.5: Metody rozhraní IClassifier

```
void train(List<TrainingData> tweet) throws IOException;
void predict(Tweet tweet);
```

8.5.1 Naivní Bayes

Naivní Bayes byl implementován bez využití knihoven. V aplikaci jsou implementovány 3 různé verze Naivního Bayese.

1. Naivní Bayes
2. Naivní Bayes s váhou slov vypočtenou pomocí tf-idf
3. Komplementární Naivní Bayes

Komplementární Naivní Bayes

Tato verze Naivního Bayese pomáhá s některými nedostatky původního algoritmu. Tento algoritmus je schopen lépe si poradit s daty, ve kterých je nevyvážený poměr hodnot pro každou třídu (např. 4 krát více negativních než pozitivních). Nevýhodou je pomalejší trénování, ale doba klasifikace zůstává stále stejná. Implementace algoritmu se nachází ve třídě `TWCNB`.

Postup při trénování algoritmu je následující:

1. Spočítat tf-idf,
2. L2 normalizace tf-idf hodnot,
3. Sečíst váhy pro jednotlivá slov přes všechny trénovací dokumenty,
4. Spočítat váhy slov pro jednotlivé třídy.

Komplementární Naivní Bayes, určuje třídu podle nejméně vhodné třídy pro vzorek. Tento postup je v programu upraven tak, aby se hledala nejvhodnější třída, jako ve standardní verzi. Výpočet váhy slova pro třídu se počítá ze všech tříd kromě té, do které prvek patří (viz ukázka 8.6).

Ukázka 8.6: Sčítání váhy slova pro neutrální vzorek dat

```
if(trainingDataList.get(i).getSentiment() == ESentiment.NEUTRAL){
   /*increment neutral class count and increase negative and
      positive weights*/
   neuClass += 1;
   posWeights.put(entry.getKey(),
      (posWeights.containsKey(entry.getKey()))?
      posWeights.get(entry.getKey()) +
      entry.getValue():entry.getValue());
   negWeights.put(entry.getKey(),
      (negWeights.containsKey(entry.getKey()))?
      negWeights.get(entry.getKey()) +
      entry.getValue():entry.getValue());
   posTotal += entry.getValue();
   negTotal += entry.getValue();
}
```

Na rozdíl od algoritmu popsaného v kapitole 4.3, je pátý krok, ve kterém je vypočten logaritmus váhy slova upraven na:

$$w_{i,j} = -log\hat{\theta}_{i,j} \tag{8.1}$$

Tato změna poté umožňuje otočit klasifikaci testovaného dokumentu a použít stejný postup klasifikace jako u NB:

$$l(t) = \underset{c}{argmax}\ t_i w_{i,j}. \tag{8.2}$$

Dalším rozdílem je vynechání kroku 6, tedy normalizace vah slova, toto rozhodnutí je dále popsáno ve výsledcích.

8.5.2 Support vector machine

Pro vytvoření SVM byla použita knihovna LibSVM[2], přesněji její přepsaná verze pro Javu. Tato knihovna umožňuje využití řídké matice, pro výpočet. Toto výrazně snižuje náročnost SVM na paměť. Postup při trénování klasifikátoru SVM:

1. vytvoření slovníku,
2. vytvoření matice obsahující jednotlivé vektory vlastností (ukázka 8.7),
3. nastavení klasifikátoru,
4. trénování klasifikátoru.

Implementace se nachází ve třídě `Supportvm`. U klasifikátoru svm je důležité správně nastavit jednotlivé parametry. Některé parametry je možné nastavit v GUI (viz kapitola 8.1).

Ukázka 8.7: Vytvoření Mapy obsující četnosti jednotlivých vlastností a řídké matice

```
for (int i = 0; i < dataCount; i++){
   tweetFeatures = cf.createFeatures(tweet.get(i).getTweet());
   indexCount= new HashMap<>();
   for (Feature feature: tweetFeatures ) {
      Integer ind = featureIndex.get(feature);
```

```
            if(ind != null){
                indexCount.put(ind, indexCount.containsKey(ind) ?
                    indexCount.get(ind) + 1 :1);
            }
        }
        int ind = 0;
        prob.x[i] = new svm_node[indexCount.size()];
        for (Map.Entry<Integer,Integer> feature: indexCount.entrySet() )
            {
            svm_node node = new svm_node();
            node.index = feature.getKey();
            node.value = feature.getValue();
            prob.x[i][ind] = node;
            ind += 1;
        }
    }
```

Při vytváření aplikace byl otestován vliv parametrů na výsledky klasifikace. Tento test slouží pouze k hledání ideálních parametrů tzv. grid-search, ale jeho průběh trvá velmi dlouho. Je možné ho zavolat z kódu: `public void gridSearch(List<TrainingData> allData) throws IOException...`. Tato metoda otestuje různé kombinace nastavení parametrů SVM. Pro každou kombinaci je proveden 10-fold test a jsou spočteny statistické ukazatele.

Klasifikace do jednotlivých tříd probíhá následujícím způsobem (ukázka 8.8):

1. vytvoření vektoru vlastností,
2. předání klasifikátoru,
3. nastavení zvoleného sentimentu testovaným datům.

Ukázka 8.8: Klasifikace tweetu do jedné ze tříd

```
int totalClasses = 3;
int[] labels = new int[totalClasses];
svm.svm_get_labels(model,labels);
double[] prob_estimates = new double[totalClasses];
double v = svm.svm_predict_probability(model,nodes,
    prob_estimates);
if(v == 2.0){
   t.setSentiment(ESentiment.POSITIVE);
}
else if(v == 0.0){
   t.setSentiment(ESentiment.NEGATIVE);
}
else if(v == 1.0){
   t.setSentiment(ESentiment.NEUTRAL);
}
```

8.5.3 Klasifikátor Maximální Entropie

Pro implementaci MaxEnt klasifikátoru je využita knihovna Brainy[9], která byla vytvořena na Západočeské Univerzitě. Pro použití klasifiktáru je nejprve nutné vytvořit jednotlivé vlastnosti, podle kterých se klasifikuje (ukázka 8.9). Generování vlastností probíhá v několika třídách, kde každá třída generuje jednu vlastnost:

- Unigramy - `Words`,
- Bigramy - `Bigrams`,
- Trigramy - `Trigrams`,
- Emotikony - `Smileys`.

Každá z těchto tříd implementuje rozhraní `Feature<T>`, kde t je v aplikaci třída `Tweet`. Každá třída tedy obsahuje následující metody: `int`

```
getNumberOfFeatures();
void extractFeature(ListIterator<T> var1, FeatureVectorGenerator
var2);
void train(InstanceList<T> var1);
```

U bigramů a trigramů je možné nastavit minimální počet kolikrát se má vlastnost vyskytovat. Samotná implementace MaxEnt klasifikátoru se nachází ve třídě `SentimentAnalyzer`.

Ukázka 8.9: Přidání jednotlivých typů vlastností do množiny

```
set = new FeatureSet<>();
set.add(new Words());
set.add(new Smileys());
set.add(new Bigrams(3));
set.add(new Trigrams(3));
```

8.6 Latentní sémantická analýza

Prvním krokem při vypočtu latentní sémantické analýze je výpočet tf-idf, o tento výpočet se stará třída `TFIDF`. Poté je vytvořena matice termů x tweet (viz ukázka 8.10).

Ukázka 8.10: Vytvoření matice term x tweet

```
for (int i = 0; i < tokens.size(); i++) {
   for (Token t: tokens.get(i)) {
      Integer q = uniqueWords.get(t);
      if( q != null) {
         double entry = tfidfvalues.get(t).get(i);
         simpleMatrix.set(q, i, entry);
      }
    }
}
```

Po vytvoření matice je provedeno SVD. Pro provedení SVD byla využita knihovna EJML (Efficient Java Matrix Library).

8.6.1 Metoda Gong a Liu

U této metody se hledá maximum (ukázka 8.11) v prvních n řádkách, jednotlivé řádky reprezentují témata obsažená v tweetech, n je počet výsledných tweetů (v aplikaci nastaven na 5). Je zde omezení, že nelze zvolit stejnou větu dvakrát.

Ukázka 8.11: Výběr vět podle metody Gong a Liu

```
for (int i = 0; i < count; i++) {
   int index = 0;
   for (int j = 0; j < vt.numCols(); j++) {
      if (!sentences.contains(j)) {
         temp = vt.get(i, j);
         if (temp > max) {
            max = temp;
            index = j;
         }
      }
   }
   sentences.add(index);
   max = Double.MIN_VALUE;
}
```

8.6.2 Metoda Steinberger a Ježek

Implementace této metody je lehce složitější než předchozí. Před výběrem věty je nejprve nutné spočítat hodnocení jednotlivých vět. Opět není možné zvolit 1 větu dvakrát.

Ukázka 8.12: Výpočet hodnocení vět

```
for (int j = 0; j < vt.numCols(); j++) {
   double tempSum = 0.0;
   for (int i = 0; i < n; i++) {
      tempSum += Math.pow(vt.get(i, j), 2) *
          Math.pow(singValues[i], 2);
   }
   tempSum = Math.sqrt(tempSum);
   sk.add(tempSum);
}
```

Po ohodnocení všech tweetů podle ukázky 8.12 je proveden výběr 5 tweetů s nejvyšším hodnocením. Tyto hodnocení jsou uloženy v `List<Double> sk`.

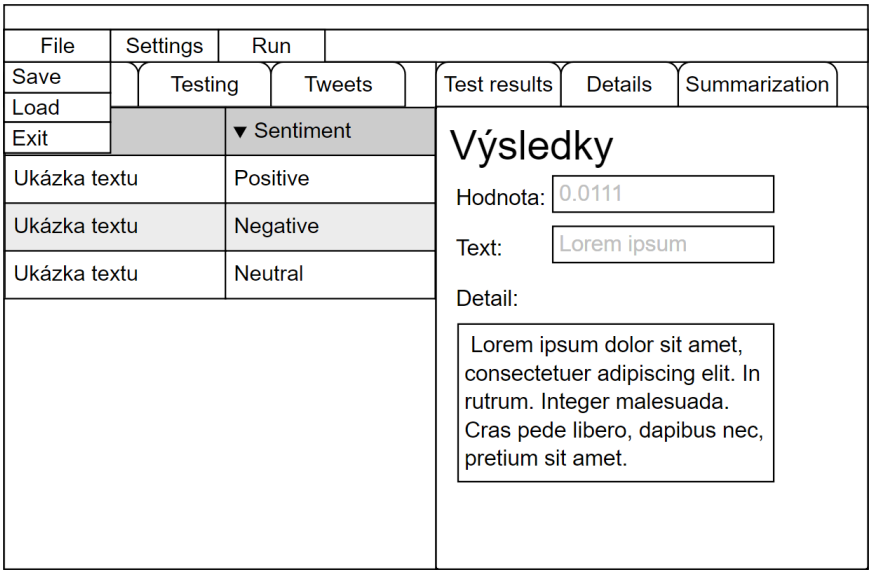
8.7 GUI

GUI je rozděleno na 2 části (viz obr. 8.2). V levé části se nachází zpracovávaná data pro trénování, testování a samotná data z Twitteru. V pravé části se poté nachází výsledky testování, sumarizace a detaily tweetu. V GUI je možné plně konfigurovat a ovládat aplikaci. Pro organizaci jednotlivých položek bylo využito záložek.

Alternativou bylo řešení, které by obrazovku dělilo vertikálně i horizontálně a tedy umožnilo více možností zobrazení, ale tento způsob byl nevhodný, kvůli jeho nutné výšce vyžadující monitor s vysokým rozlišením.

V aplikaci je možné změnit velikost textu aplikace, nastavit různé barvy jednotlivých částí textu tweetů, veškeré nastavení klasifikátorů a testování.

Vertikálně je aplikace rozdělena na dvě části, horní část obsahující menu (`MenuBar`) a spodní část hlavní okno (`HBox`). K tomuto rozdělení byl využit `VBox`.



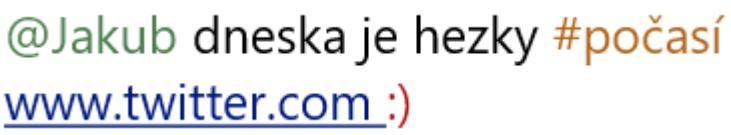
Obrázek 8.2: Návrh vzhledu aplikace

Pro vytvoření záložek je využit `TabPane`, do kterého jsou poté vloženy jednotlivé záložky `Tab`.

8.7.1 Buňka tabulky pro text tweetu

Jedná se o jednotlivé buňky sloupce „Text" ve všech tabulkách implementace je ve třídě `TweetCell`. Standardní buňka umožňuje zobrazení textu v jedné barvě. Pro lepší zobrazení, byla vytvořena tato buňka, která umožňuje zobrazit různé části textu různou barvou (viz obr. 8.3). Barvu je možné nastavit v konfiguračním souboru nebo v dialogu s nastavením. Pro vytvoření vlastní buňky musí třída dědit od `TableCell<T,E>`, v tomto případě je E nastaveno na `String` a T může být `Tweet` nebo `TrainingData`.

JavaFX v základu nepodporuje změnu jednotlivých částí textu je pouze možné nastavit barvu textu v CSS. Ukázka nastavení barvy textu: `component.setStyle("-fx-text-fill: blue;")`.
Tímto způsobem je možné změnit pouze celý text, a proto toto řešení není



Obrázek 8.3: Ukázka buňky s barevným textem

vhodné. Pro vytvoření požadovaného efektu je nutné využít třídy `Text`, která umožňuje nastavit barvu, velikost textu atd. (ukázka 8.13) Poté bylo nutné jednotlivé části textu uložit tak, aby se ve výsledku zobrazily jako normální text oddělený mezerami, k tomu bylo využito komponenty `TextFlow`, která umožňuje ukládat různé texty a zobrazit je.

Ukázka 8.13: Vytvoření jedné části textu, nastavení barvy a velikosti textu

```
TokenType type = item.get(i).getType();
Color color = getTokenType(type);
String space = (i == size-1) ? "" : " ";
Text t = new Text(item.get(i).getText() + space);
text += item.get(i).getText() + space;
t.setFill(color);
if(type == TokenType.url){
   t.setUnderline(true);
}
t.setFont(Font.font(t.getFont().getFamily(),
      Settings.getInstance().getTextSize()));
```

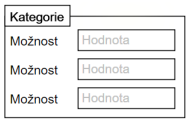
Výšku řádku tabulky je nutné nastavit podle délky textu, pokud by nebyla výška nastavena, nebyl by text, který je delší než 1 řádka zobrazen celý. Problémem je, že `TextFlow` nevrací svou přesnou výšku, je tedy nutné tento rozměr spočítat a nastavit ho buňce (ukázka 8.14). Pro zjištění délky textu je použita třída `Text`, která umožňuje získat šířku kompletního textu. Po celou dobu běhu aplikace se ukládá šířka sloupce, do kterého je buňka vložena. Takže lze získat požadovanou výšku a nastavit ji dané buňce. Před výpočte je nutné textu nastavit správnou velikost, jinak by výpočet nedával smysl.

Ukázka 8.14: Nastavení velikosti pro buňku obsahující text

```
double rows = Math.ceil(a / (getColumnWidth()-10));
height = Math.ceil((t.getLayoutBounds().getHeight()))* rows +
    gap*(rows -1);
tweetDisplay1.setMinHeight(height);
tweetDisplay1.setPrefHeight(height);
tweetDisplay1.setMaxHeight(height);
```

8.7.2 Dialog nastavení

Toto modální dialogové okno je implementováno ve třídě `SettingsDialog` pro vytvoření vlastního dialogového okna je nutné dědit od třídy `Dialog<T>`, T je v aplikaci `Settings`. Tento dialog umožňuje nastavit všechny dostupné možnosti aplikace. Všechny nastavitelné položky jsou popsány v oddílu 8.1.



Obrázek 8.4: Návrh zobrazení kategorie možností

Pro oddělení jednotlivých kategorií možností byl původně navržen vzhled, který by ohraničoval kategorii a v rohu by byl název kategorie (viz obr. 8.4). Takovéto zobrazení JavaFX neumožňuje, bylo proto využito jiné komponenty, kterou lze upravit do podobného zobrazení. Touto komponentou je `TitledPane`, v základní verzi se jedná o komponentu jejichž obsah je možné rozbalit/sbalit. V řešení aplikace je tato funkcionalito vypnuta a výsledný vzhled je možné vidět na obr. 8.5.

Dialog obsahuje některé položky `TextField`, které vyžadují číslo (`int`, `double`, ...), těmto položkám je nutné nastavit `TextFormatter`, který kontroluje tvar vkládaných znaků. Pro vytvoření `TextFormatter` je nutné nejprve implementovat filtr (viz ukázka 8.15) a konvertor (viz ukázka 8.16). Po na-

stavení `TextFormatter` danému textovému poli není možné vložit do tohoto pole jinou hodnotu než, která je povolena. Příkladem položky, která je takto omezena je například hodnota hyperparametru SVM C.

Ukázka 8.15: Vytvoření filtru pro datový typ double

```
private Pattern validEditingState =
    Pattern.compile("(([1-9][0-9]*)|0)?(\\.[0-9]*)?");
UnaryOperator<TextFormatter.Change> filter = c -> {
   String text = c.getControlNewText();
   if (validEditingState.matcher(text).matches()) {
      return c ;
   } else {
      return null ;
   }
};
```

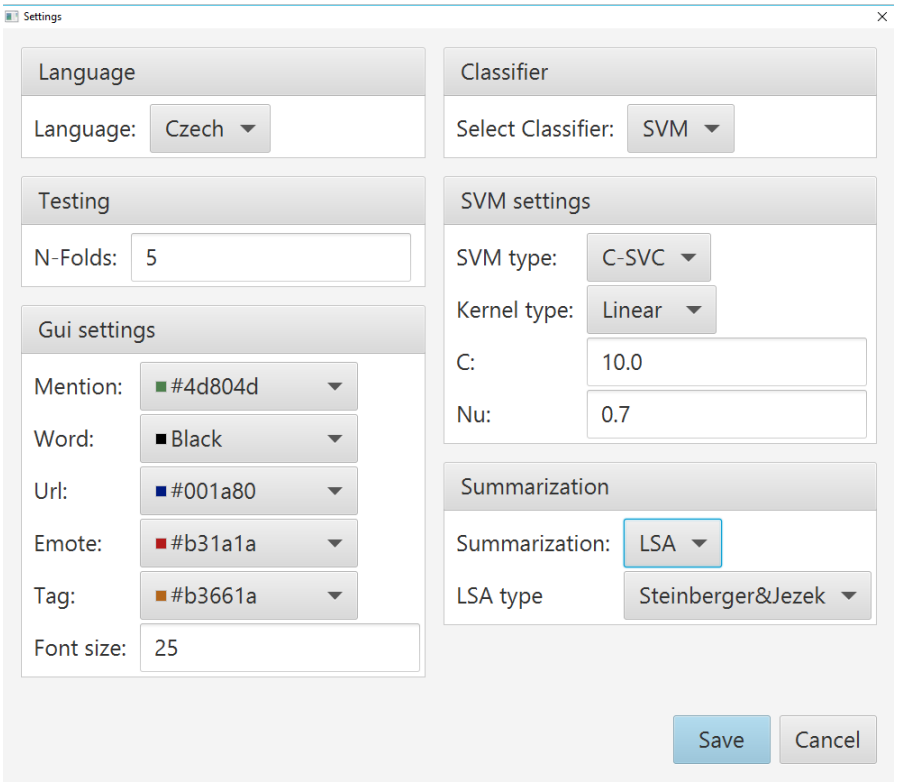
Ukázka 8.16: Vytvoření konvertoru pro datový typ double

```
StringConverter<Double> converter = new StringConverter<Double>() {

   @Override
   public Double fromString(String s) {
      if (s.isEmpty() || "-".equals(s) || ".".equals(s) ||
          "-.".equals(s)) {
         return 0.0 ;
      } else {
         return Double.valueOf(s);
      }
   }

   @Override
   public String toString(Double d) {
      return d.toString();
   }
};
```

Před odesláním formuláře jsou validovány jednotlivé parametry, pokud je detekována neplatná hodnota, je uživateli tato skutečnost oznámena informačním dialogem. Pokud validace projde v pořádku proběhne změna nastavení.



Obrázek 8.5: Kompletní dialog nastavení

9 Výsledky

Aplikace byla testována na několika datasetech. Pro sentiment jsou využity následující datasety:

1. Facebook CZ [7],
2. DAI Twitter dataset [1].

Dataset z českého Facebooku obsahuje 2 587 pozitivních, 5 174 neutrálních, 1 991 negativních a 248 bipolárních příspěvků. Bipolárním příspěvkům je při trénování přidělen neutrální sentiment.

Druhý dataset obsahuje data v angličtině převzatá ze sociální sítě Twitter. Dataset byl ohodnocen třemi lidmi. Proto dataset obsahuje několik verzí, podle toho kolik shod bylo při hodnocení dosaženo. Použitá byla verze se třemi shodami. Tento dataset obsahuje 3771 příspěvků. Z těchto je 488 negativních, 739 pozitivních a 2536 neutrálních. Pro porovnání dataset, který bere v úvahu všechny možné hodnocení bez ohledu na shodu obsahuje 21600 záznamů, pro každou větu jsou 3 různá hodnocení. Z těchto hodnocení je 5228 pozitivních, 12538 neutrálních a 3254 negativních. Dataset byl dále modifikován odstraněním nepotřebných sloupců, aby bylo možné ho v aplikaci použít. Dataset je tedy ve tvaru „sentiment text".

Testovány byly následující klasifikátory:

1. Transformed Complement Naive Bayes (TCNB)
2. Naivní Bayes - TF-IDF,
3. SVM,
4. Maximální entropie.

[1] `http://dainas.aot.tu-berlin.de/~andreas@dai/sentiment/`

Pro každý klasifikátor je spočítána přesnost (precision) vz. 9.1, úplnost (recall) vz. 9.2 a f-míra (f-measure) vz. 9.3. TP (true positive) jsou prvky, které byly klasifikovány správně. FP (false positive) byly klasifikovány, jako pozitivní, ale patří do jiné třídy. FN (false negative) prvky, které patří do požadované třídy, ale klasifikátor je nerozeznal. U sentimentu jsou tyto ukazatele spočítány pro každou třídu (pozitivní, negativní, neutrální) zvlášť a také je spočítána celková hodnota.

$$Precision = \frac{TP}{TP + FP} \tag{9.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{9.2}$$

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{9.3}$$

Existují dva způsoby výpočtu průměrných hodnot každé z metrik:

- mikro průměr (micro average),
- makro průměr (macro average).

Mikro průměr se počítá podle:

$$Micro - Precision = \frac{TP_1 + \cdots + TP_n}{TP_1 + \cdots + TP_n + FP_1 + \cdots + FP_n} \tag{9.4}$$

$$Micro - Recall = \frac{TP_1 + \cdots + TP_n}{TP_1 + \cdots + TP_n + FN_1 + \cdots + FN_n} \tag{9.5}$$

F-míra se pro mikro průměr počítá stejně jako u neprůměrovaných hodnot.

Makro průměr vypočte jednotlivé ukazatele pro každou metriku a poté vypočítá průměrnou hodnotu. Výpočet f-míry se ani v tomto případě nemění.

Vzorce jsou tedy následující (Pr je Přesnost a Re je úplnost):

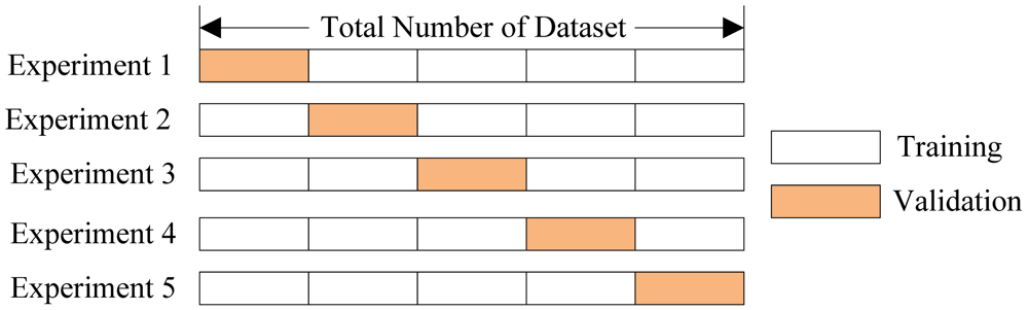
$$Macro - Precision = \frac{Pr_1 + Pr_2 + \cdots + Pr_n}{n} \tag{9.6}$$

$$Macro - Recall = \frac{Re_1 + Re_2 + \cdots + Re_n}{n} \tag{9.7}$$

Pro samotné hodnocení klasifikátorů sentimentu byla zvolena makro průměrná hodnota jednotlivých metrik.

9.1 Sentiment

Pro testování sentimentu je použita křížová validace. Pro každý klasifikátor je tedy 10krát spuštěno trénování a klasifikace na vždy odlišných datech (viz obr. 9.1). Počet kolikrát se změní data je v aplikaci nastaven na 10.



Obrázek 9.1: Ukázka křížové validace. Zdroj[2]

9.1.1 Testování na českých datech

Při testování dat z českého Facebooku je pro trénování využito 9000 zpráv a na 1000 bylo testováno. Z výsledků (tab. 9.1) lze vyvodit celkem výrazné

[2]https://www.kaggle.com/dansbecker/cross-validation

Tabulka 9.1: Výsledky testování klasifikátorů na datech z českého Facebooku

Klasif. / Měření	TCNB	NB TFIDF	MaxEnt	SVM
Macro pozitivní přesnost	0,6296	0,6808	0,7621	0,7123
Macro neutrální přesnost	0,7566	0,7332	0,6856	0,5808
Macro negativní přesnost	0,5226	0,5749	0,5992	0,6428
Macro celková přesnost	0,6363	0,6630	0,6823	0,6453
Macro pozitivní úplnost	0,8109	0,7411	0,6178	0,3110
Macro neutrální úplnost	0,6684	0,7518	0,8527	0,9499
Macro negativní úplnost	0,5006	0,4710	0,3790	0,0637
Macro celková úplnost	0,6600	0,6547	0,6165	0,4415
Macro pozitivní f-míra	0,7081	0,7093	0,6810	0,4312
Macro neutrální f-míra	0,7084	0,7414	0,7584	0,7199
Macro negativní f-míra	0,5099	0,5165	0,4591	0,1149
Macro celková f-míra	0,6421	0,6558	0,6328	0,4220

nedostatky při klasifikaci negativních tweetů. Toto by mohlo být způsobeno výskytem smajlíků s pozitivním míněním v negativních větách. Ze zvolených klasifikátorů se všechny kromě SVM dostaly nad 0,63 f-míry. SVM vykazovalo největší nedostatky při testování a proto bych SVM ze zvolených metod nedoporučil.

Zajimvé jsou výsledky obou testovaných verzí algoritmu Naivního Bayese, který v testu dosáhl nejlepších výsledků. Algoritmus maximální entropie sice dosahuje v průměru nejlepší přesnosti, ale v úplnosti zaostává za NB.

9.1.2 Testování na anglickém Twitteru

Při testování bylo opět využito 90% dat na trénování a 10% na testování. V tomto testu dosáhly klasifikátory srovnatelných výsledků (tab. 9.2). I u těchto dat došlo k nevyváženosti jednotlivých tříd, a proto je možné, že dochází k tomuto znepřesnění. Nejlépe dopadl komplementární naivní bayes, který na tomto datasetu dosáhl mírně lepších výsledků než MaxEnt a SVM. Výrazný je vzestup SVM na tomto datasetu, vzhledem k jeho nízké úspěš-

Tabulka 9.2: Výsledky testování klasifikátorů na datech z anglického Twitteru

Klasif. / **Měření**	**TCNB**	**NB TFIDF**	**MaxEnt**	**SVM**
Macro pozitivní přesnost	0,5663	0,5882	0,5500	0,7300
Macro neutrální přesnost	0,9193	0,9005	0,8755	0,7180
Macro negativní přesnost	0,6000	0,5135	0,7021	0,5071
Macro celková přesnost	0,6952	0,6674	0,7092	0,6093
Macro pozitivní úplnost	0,7015	0,6898	0,6567	0,1856
Macro neutrální úplnost	0,8039	0,7943	0,8549	0,9727
Macro negativní úplnost	0,7778	0,6935	0,6111	0,1438
Macro celková úplnost	0,7611	0,7259	0,7076	0,7101
Macro pozitivní f-míra	0,6267	0,6341	0,5986	0,2931
Macro neutrální f-míra	0,8577	0,8436	0,8651	0,8260
Macro negativní f-míra	0,6774	0,5852	0,6535	0,2185
Macro celková f-míra	0,7206	0,6876	0,7057	0,7101

nosti na českých datech. SVM vzkazovalo největší nedostatky v klasifikci pozitivních dat, kde dosáhlo velmi nízké úplnosti 0,1856

9.1.3 Porovnání Komplementárního Naivního Bayese s/bez normalizace

Podle výsledků z tab. 9.3 je vidět, že normalizace výrazně ovlivnila úplnost negativních dat, která se snížila o 0,2285. Vzhledem k tomu, že aplikace vrací výsledky podle kladných a záporných dat, je důležité získat výsledky právě v těchto kategoriích. Proto byla v práci použita verze bez normalizace.

9.1.4 Vliv vlastností na MaxEnt

V tabulce 9.4 je naznačen vliv jednotlivých vlastností na klasifikátor MaxEnt. Tabulka ukazuje jaký vliv na klasifikaci má postupné přidávání jednotlivých vlastností. Z tabulky je vidět, že největší vliv mají bigramy a emo-

Tabulka 9.3: Porovnání CNB s a bez normalizace na datech z českého Facebooku

Normalizace. / **Měření**	**Ano**	**Ne**
Macro pozitivní přesnost	0,6296	0,6919
Macro neutrální přesnost	0,7566	0,6965
Macro negativní přesnost	0,5226	0,6712
Macro pozitivní úplnost	0,8109	0,7208
Macro neutrální úplnost	0,6684	0,8458
Macro negativní úplnost	0,5006	0,2715
Macro pozitivní f-míra	0,7081	0,7057
Macro neutrální f-míra	0,7084	0,7629
Macro negativní f-míra	0,5099	0,3841

tikony. U trigramů je ve výsledcích vidět minimální rozdíl proti bigramům. Je zajímavé, že při využití pouze unigramů je dosaženo nejvyšší úplnosti klasifikace negativních dat, ale na druhou stranu i nejnižší přesnosti.

Tabulka 9.4: Výsledky testování vlivu vlastností na MaxEnt klasifikátor,

Vlastnosti / **Měření**	**Unig.**	**+ Bigr**	**+ Trigr.**	**+ Emot.**
Macro poz. přesnost	0,5614	0,6113	0,6299	0,6720
Macro neu. přesnost	0,8418	0,8412	0,8388	0,8463
Macro neg. přesnost	0,5263	0,5629	0,5557	0,5955
Macro poz. úplnost	0,5891	0,6140	0,6140	0,6440
Macro neut. úplnost	0,8242	0,8487	0,8564	0,8753
Macro neg. úplnost	0,5419	0,5334	0,5199	0,5273
Macro pozit. f-míra	0,5733	0,6114	0,6208	0,6556
Macro neut. f-míra	0,8327	0,8487	0,8472	0,8603
Macro neg. f-míra	0,5313	0,5453	0,5339	0,5562

9.2 Sumarizace

Sumarizace je otestována metodou ROUGE [10] (Recall-Oriented Understudy for Gisting Evaluation), přesněji ROUGE-1. Metoda ROUGE-1 je vypočtena podle vzorce 9.8. V tomto vzorci je pocet$_s$*hodne*(gram$_n$) pocet n-gramů které se shodují ve vytvořené sumarizaci s kontrolní, pocet(gram$_n$) je celkový počet n-gramů kontrolní sumarizace a n specifikuje řád n-gramu (1 = unigram, 2 = bigram...). Z tohoto vzorce je vidět, že se jedná o úplnost.

$$ROUGE - N = \frac{pocet_{shodne}(gram_n)}{pocet(gram_n)} \tag{9.8}$$

Sumarizace je testována na anglickém datasetu Opiniosis opinion dataset 1,0 [5]. Tento dataset obsahuje 51 témat pro sumarizaci. Každé téma obsahuje kolem 100 vět a pro každé je dostupné kole 4 člověkem vytvořených sumarizací. Data jsou získána z různých zdrojů: Tripadvisor (hotely), Edmunds.com (auta) a Amazon.com (různá elektronika).

Vzhledem k tomu, že použitá metoda sumarizace je generativní a porovná se s abstraktní cílovou sumarizací mohou být výsledky nižší než by měly být. Z tabulky 9.5 je vidět, že metoda podle Steinbergera a Ježka vrací sumarizaci, která je v průměru podobnější cílovým sumarizacím. Nelze však prohlásit, že rozdíl je na testovaném datasetu výrazný.

Tabulka 9.5: Výsledek otestované sumarizace metodou ROUGE-1

Verze / **Měření**	**Gong a Liu**	**Steinberger a Ježek**
Průměrný ROUGE-1	0,3815	0,3928

10 Závěr

Cílem práce bylo vytvořit prostředek pro analýzu Twitterových příspěvků, s jehož pomocí by bylo možné zjišťovat agregované názory přispěvatelů.

V práci byly představeny sociální sítě a byl zmíněn jejich význam a možné využití v současné společnosti. Dále byla popsána sociální síť Twitter a způsob přístupu k datům a jejich formát.

Také byly vysvětleny použité metody strojového učení a předzpracování textu. Byly popsány metody pro sumarizaci textu a rozdíly mezi typy sumarizace.

V hlavní části práce je poté popsáno samotné řešení a výsledky. U každé ze zvolených metod klasifikace sentimentu je popsán postup, jak fungují. U výsledků sentimentu na českých datech je zajímavá nízká úspěšnost SVM pro klasifikaci sentimentu. Jako nejlepší z implementovaných se ukázal Komplementární Naivní Bayes, který je velmi těsně následován klasifikátorem MaxEnt a NB s TF-IDF.

Aplikaci by bylo možné dále rozšířit o nové vlastnosti např. POS (part of speech) tagy, rozpoznávání pojmenovaných entit.... Dále by bylo možné přidat další klasifikátory pro porovnání se současnými.

K aplikaci bylo vytvořeno GUI, které umožňuje aplikaci plně ovládat. Vzhledem k tomu, že GUI bylo vytvořeno ke konci vývoje programu neobsahuje některé funkce, které by zvýšily jeho použitelnost (např progress bary atd.).

I když aplikaci měla být napsána pro češtinu, je možné jí rozšířit o podporu dalších jazyků bez výrazných změn. V současné době podporuje navíc angličtinu.

Pro vyhledávání dat na Twitteru se po několika pokusech ukázalo jako

nejlepší používat tagy a lokaci, které vysoce zlepšují kvalitu nalezených tweetů, obzvlášť co se týče jazyka.

Při testování aplikace na českém Twitteru (viz příloha C) byly zjištěny dobré výsledky i přesto, že nebyl program pro danou tématiku natrénován. I na trénovacích datech byly zjištěné dobré výsledky na českých i anglických datech.

Seznam zkratek

API - Aplication programming interface

REST - Representational State Transfer

NB - Naivní Bayes

SVM - Supprt Vector Machine

TF-IDF - Term frequency - inverse document frequency

MaxEnt - Maximum entropy

LSA - Latentní sémantická analýza

SVD - Singular value decomposition

TWNCNB - Transformed Weignt-normalized Complement Naive Bayes

HPS - High Precision Stemmer

EJML - Efficient Java Matrix Library

CSS - Cascade Style Sheet

ROUGE - Recall-Oriented Understudy for Gisting Evaluation

POS - Part of Speech

Seznam obrázků

Seznam tabulek

Bibliografie

[1] Tomáš Brychcín a Miloslav Konopík. "HPS: High precision stemmer". In: *Information Processing & Management* 51.1 (2015), s. 68 –91. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2014.08.006. URL: http://www.sciencedirect.com/science/article/pii/S0306457314000843.

[2] Chih-Chung Chang a Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

[3] James H. Martin Daniel Jurafsky. "Speech and Language Processing. An Intorduction to Natural Language Processing, Computational Linguistics and Speech recognition". In: Prentice Hall, 2008. ISBN: 978-0131873216.

[4] *Dokumentace API Twitter [online]*. Dostupné z: https://developer.twitter.com/en/docs. [cit. 6-4-2018].

[5] Kavita Ganesan, ChengXiang Zhai a Jiawei Han. "Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions". In: *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics. 2010, s. 340–348.

[6] Yihong Gong a Xin Liu. "Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis". In: *Proceedings of the*

24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '01. New Orleans, Louisiana, USA: ACM, 2001, s. 19–25. ISBN: 1-58113-331-6. DOI: 10.1145/383952.383955. URL: http://doi.acm.org/10.1145/383952.383955.

[7] Ivan Habernal, Tomáš Ptácek a Josef Steinberger. "Sentiment analysis in czech social media using supervised machine learning". In: *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 2013, s. 65–74.

[8] Dan Jurafsky a James H Martin. *Speech and language processing*. Sv. 3. Pearson London: 2014.

[9] Michal Konkol. "Brainy: A Machine Learning Library". In: *Artificial Intelligence and Soft Computing*. Ed. Leszek Rutkowski et al. Sv. 8468. Lecture Notes in Computer Science. Springer International Publishing, 2014, s. 490–499. ISBN: 978-3-319-07175-6. DOI: 10.1007/978-3-319-07176-3_43. URL: http://dx.doi.org/10.1007/978-3-319-07176-3_43.

[10] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of summaries". In: *Proceedings of the ACL Workshop: Text Summarization Braches Out 2004* (led. 2004), s. 74–81.

[11] Bing Liu. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

[12] Sascha Narr, Michael Hulfenhaus a Sahin Albayrak. "Language-independent twitter sentiment analysis". In: *Knowledge discovery and machine learning (KDML), LWA* (2012), s. 12–14.

[13] Alexander Pak a Patrick Paroubek. "Twitter as a corpus for sentiment analysis and opinion mining." In: 10.2010 (2010).

[14] Bo Pang, Lillian Lee a Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques". In: *Proceedings of the ACL-02 conference on Empirical methods in natural language*

processing-Volume 10. Association for Computational Linguistics. 2002, s. 79–86.

[15] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries". In: 242 (2003), s. 133–142.

[16] *Registrace aplikací [online]*. Dostupné z: `https://apps.twitter.com/`. [cit. 6-4-2018].

[17] Jason D. M. Rennie et al. "Tackling the Poor Assumptions of Naive Bayes Text Classifiers". In: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. ICML'03. Washington, DC, USA: AAAI Press, 2003, s. 616–623. ISBN: 1-57735-189-4. URL: `http://dl.acm.org/citation.cfm?id=3041838.3041916`.

[18] Shai Shalev-Shwartz a Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. ISBN: 1107057132.

[19] Josef Steinberger a Karel Jezek. "Using latent semantic analysis in text summarization and summary evaluation". In: *Proc. ISIM* 4 (2004), s. 93–100.

[20] Vladimir Naumovich Vapnik. "An overview of statistical learning theory". In: *IEEE transactions on neural networks* 10.5 (1999), s. 988–999. ISSN: 1045-9227.

Příloha A: Uživatelská příručka

Pro přeložení a spuštění programu je potřeba Java 1.8 a Maven 2.3 a vyšší. Program je možné přeložit dvěma způsoby (pro Windows), pro Linux je nutné využít příkazové řádky:

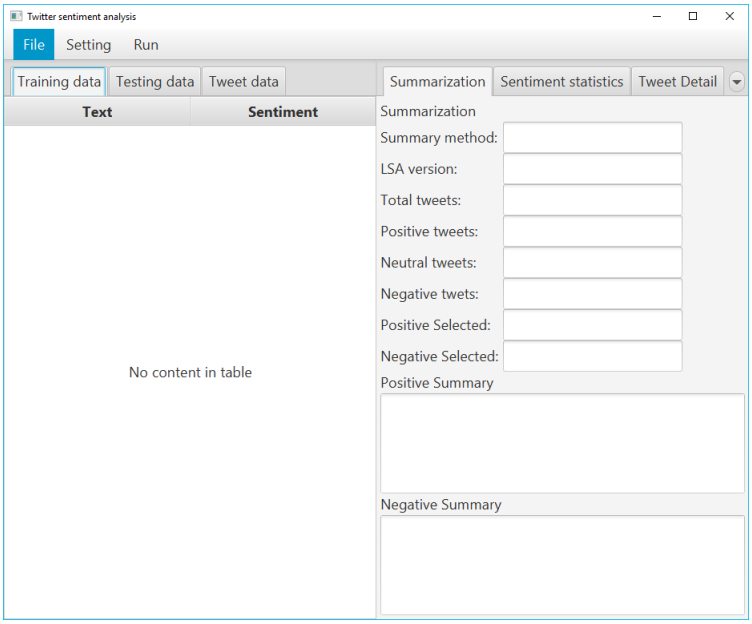
- Dávkový soubor `createwindowsjar.bat`,
- Z příkazové řádky.

Pokud byla zvolena možnost dva, je nutno zadat následující příkazy pro přeložení (je možné, že bude nutné nastavit proměnné prostředí pro Maven, někdy se používá „maven" místo „mvn"):

```
mvn clean
mvn install:install-file -Dfile=lib/Brainy-0.6.5.jar
-DgroupId=cz.zcu.kiv.nlp -DartifactId=Brainy
-Dversion=0.6.5 -Dpackaging=jar
mvn com.zenjava:javafx-maven-plugin:8.8.3:jar
```

Při provádění těchto příkazů by neměla bý být zobrazena žádná chyba. Po provedení těchto příkazů se vytvoří nový `jar` soubor, který se nachází ve složce `twittersentimentanalysis/target/jfx/app/`.

Pokud byla zvolena první možnost nachází se přeložený soubor ve formátu `jar` ve složce s názvem `jar`. Jedná se o spustitelný soubor, takže je možné ho spustit jako standardní aplikaci nebo z příkazové řádky příkazem:



Obrázek 10.1: Program po spuštění

```
java -jar <nazev_jar>
```

Příklad:

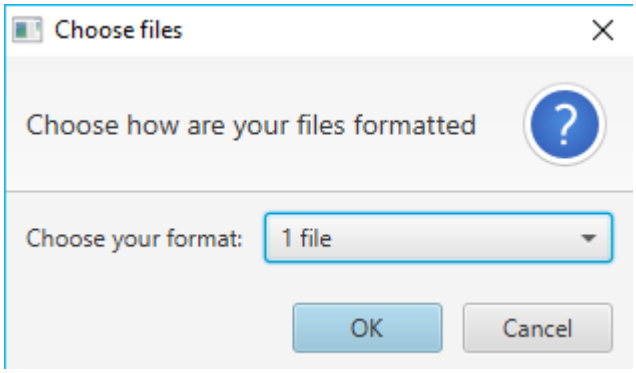
```
java -jar twittersentimentanalysis-1.0-SNAPSHOT-jfx.jar
```

Pokud se aplikace úspěšně spustila, bude zobrazena základní obrazovka (obr 10.1).

Nastavení programu

Nastavení bylo již zmíněno v oddílu 8.1. Po provedení změn je uživatel vyzván k potvrzení změn, pokud jsou všechny nově nastavené hodnoty v pořádku, nastaví se nové hodnoty a okno nastavení se ukončí.

Po provedení změn jazyka a nastavení klasifikátoru, je nutné znovu natrénovat daný klasifikátor.



Obrázek 10.2: Výběr formátu souborů

Načtení trénovacího souboru

Trénovací soubor je možné načíst volbou v menu `File->Load Training File`. Po zvolení této položky se zobrazí menu pro výběr typu souborů, na kterých se bude trénovat (obr 10.2). Existují 2 možnosti:

- 1 soubor obsahující sentiment i text (XML, JSON, CSV, TXT) - možnost `1 file`,
- 2 soubory pro sentiment a pro text (TXT) - možnost `TXT 2 files (labels, texts)`.

1 Soubor

Pokud byla zvolena první možnost, je uživateli zobrazeno standardní okno pro výběr souboru. Nejprve je nutné zvolit formát, v jakém jsou načítaná data, v pravém dolním rohu zobrazeného dialogového okna. Poté je nutné zvolit požadovaný soubor a otevřít. Pokud vše proběhne v pořádku jsou načtená data zobrazena v záložce `Training data`.

2 soubory pro sentiment a pro text

Při zvolení této varianty je potřeba zvolit 2 soubory v určeném pořadí:

1. Soubor se sentimentem (pozitivní, negativní, neutrální),
2. Soubor s texty příspěvků.

Pokud vše proběhne v pořádku jsou načtená data zobrazena v záložce `Training data`.

Načtení testovacího souboru

Pro načtení trénovacích dat je volba v menu `File->Load Testing File`. Načtení testovacího souboru probíhá stejně, jako načtení trénovacího souboru. Jediný rozdíl je zobrazení dat v záložce `Testing data`.

Načtení tweetů ze souboru

Tweety pro klasifikaci a sumarizaci je možné načíst pouze ze souboru ve formátu JSON. Načtení je možné provést volbou `File->Load Tweets File`. Po zvolení je zobrazen dialog pro výběr souboru obsahující tweety. Po úspěšném otevření jsou data zobrazena v tabulce `Tweet data`.

Načtení klasifikátoru

Pro načtení klasifikátoru ze souboru je volba `File->Load Classifier`. Po zvolení je zobrazen dialog pro otevření souboru. Důležité je načítat pouze

klasifikátor, který je aktuálně nastavený, jinak se zobrazí dialogové okno s chybou, které vyzývá k napravení tohoto problému.

Vyhledávání tweetů na Twitteru

Pro vyhledávání přímo na Twitteru je možnost `File->Search tweets`. Tato volba vyvolá dialogové okno (obr 10.3), které umožňuje nastavit parametry pro vyhledávání na Twitteru:

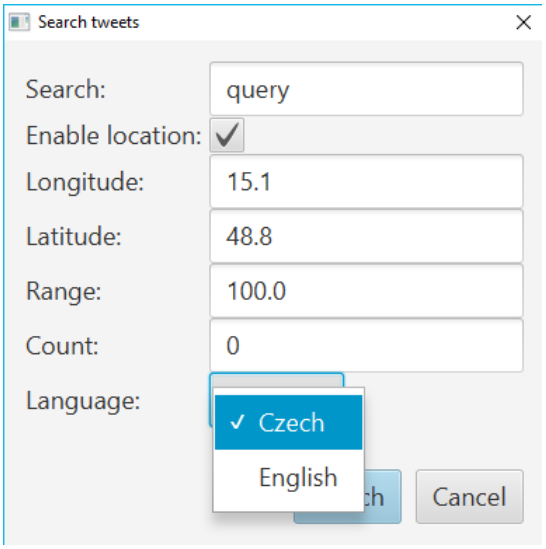
- Search - Dotaz na Twitter (např. worldcup),
- Enable location - Zda se má použít vyhledávání podle lokace,
- Longitude, Latitude - Zeměpisná délka, šířka,
- Range - Poloměr kružnice, ve které se bude vyhledávat v kilometrech,
- Language - Jazyk, ve kterém bude provedeno vyhledávání,
- Count - Počet kolik se má vyhledat tweetů (výchozí hodnota je 500).

Pokud je count nastaven na 0 je použita výchozí hodnota 500. Po provedení dotazu jsou načtena odpovídající data a je zobrazen počet nalezených tweetů.

Ukládání trénovacích, testovacích, Twitterových dat a výsledků sumarizace

Jedná se o tyto 4 volby:

- `File->Save Training File`,
- `File->Save Testing File`,



Obrázek 10.3: Dialog pro vytvoření dotazu na Twitter

- `File->Save Tweets File`.
- `File->Save Summarization`.

Pokud existují data k uložení, je zobrazen dialog pro výběr místa uložení. Pokud ukládání proběhne v pořádku, je vytvořen JSON soubor obsahující uložené informace.

Ukládání natrénovaného klasifikátoru

Volba `File->Save Classifier`. Pokud klasifikátor, který je momentálně nastavený v nastavení programu je dostupný (byl před uložením natrénován), je zobrazeno okno pro výběr místa uložení. Po uložení je vytvořen soubor, který je možné později znovu načíst viz výše.

Trénování klasifikátoru

Pro natrénování klasifikátoru je nejprve nutné načíst trénovací data. Pokud jsou data načtena, je možné spustit trénování. Trénování je spuštěno volbou `Train Classificator`. Tato operace může chvíli trvat v závislosti na velikosti dat. Po natrénování se zobrazí dialog upozorňující uživatele na úspěšné nebo neúspěšné dokončení trénování.

Klasifikace tweetů

Pro klasifikaci tweetů je potřeba natrénovaný klasifikátor a tweety určené ke klasifikaci (data v tabulce `Tweet data`). Klasifikace je spuštěna volbou `Run->Run Classification`. Po klasifikaci se zobrazí dialog upozorňující uživatele na úspěšné nebo neúspěšné dokončení klasifikace. Po dokončení je sentiment zobrazen v tabulce `Tweet data`.

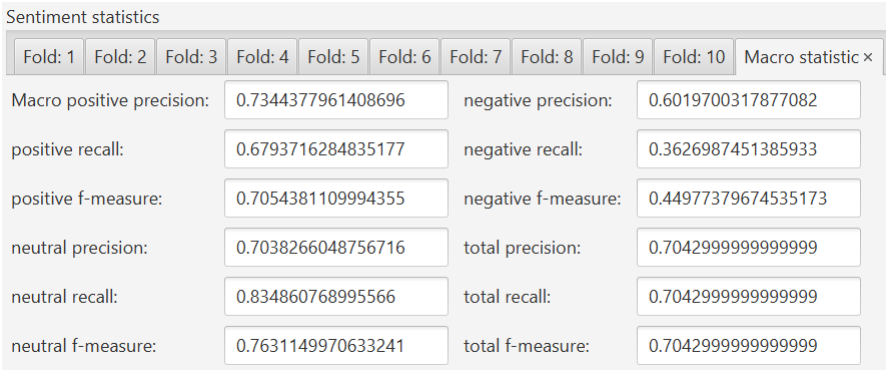
Testování klasifikátoru

Aplikace umožňuje 2 způsoby testování:

1. Nfold test - `Run->Run nfold test`,
2. Testování s testovacím souborem `Run->Run test`.

Nfold test

Pro tento test jsou potřeba pouze trénovací data. Tato operace může trvat delší dobu, protože program několikrát provede natrénování, klasifikaci a výpočet statistik. Po dokončení jsou výsledky zobrazeny v záložce `Sentiment statistics` (viz obr. 10.4).



Obrázek 10.4: Výsledky n-fold testu

Testování s testovacím souborem

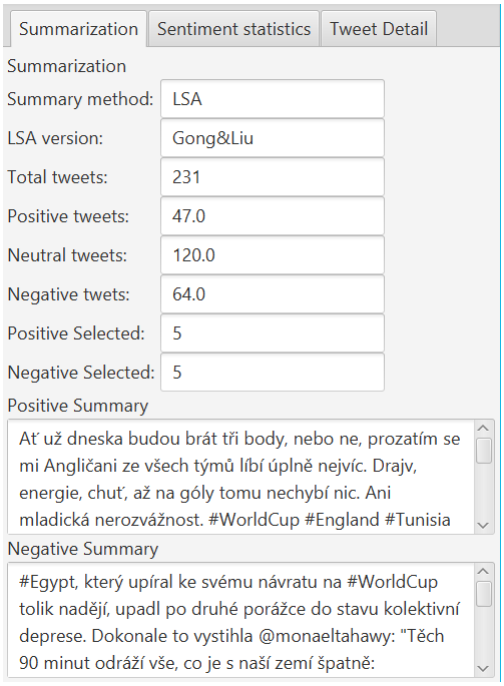
Pro tento test jsou potřeba trénovací data a testovací data. Po dokončení jsou výsledky zobrazeny v záložce `Sentiment statistics` (viz obr. 10.4).

Provedení sumarizace

Sumarizaci je možné provést na ohodnocených datech. Sumarizace je spuštěna volbou `Run->Run Summarization`. Po dokončení je výsledek zobrazen v záložce `Summarization`, která obsahuje jednotlivé sumarizace, počet negativních, neutrálních a pozitivních tweetů a počet vybraných tweetů pro sumarizaci (viz obr. 10.5).

Testování sumarizace

Pokud je ve složce na stejné úrovni dostupný Opiniosis opinion dataset, je možné spustit test sumarizace textu pro metodu zvolenou v nastavení aplikace. Test sumarizace je spuštěn voláním `Run->Test Summarization`. Po dokončení je zobrazen dialog s výsledky.



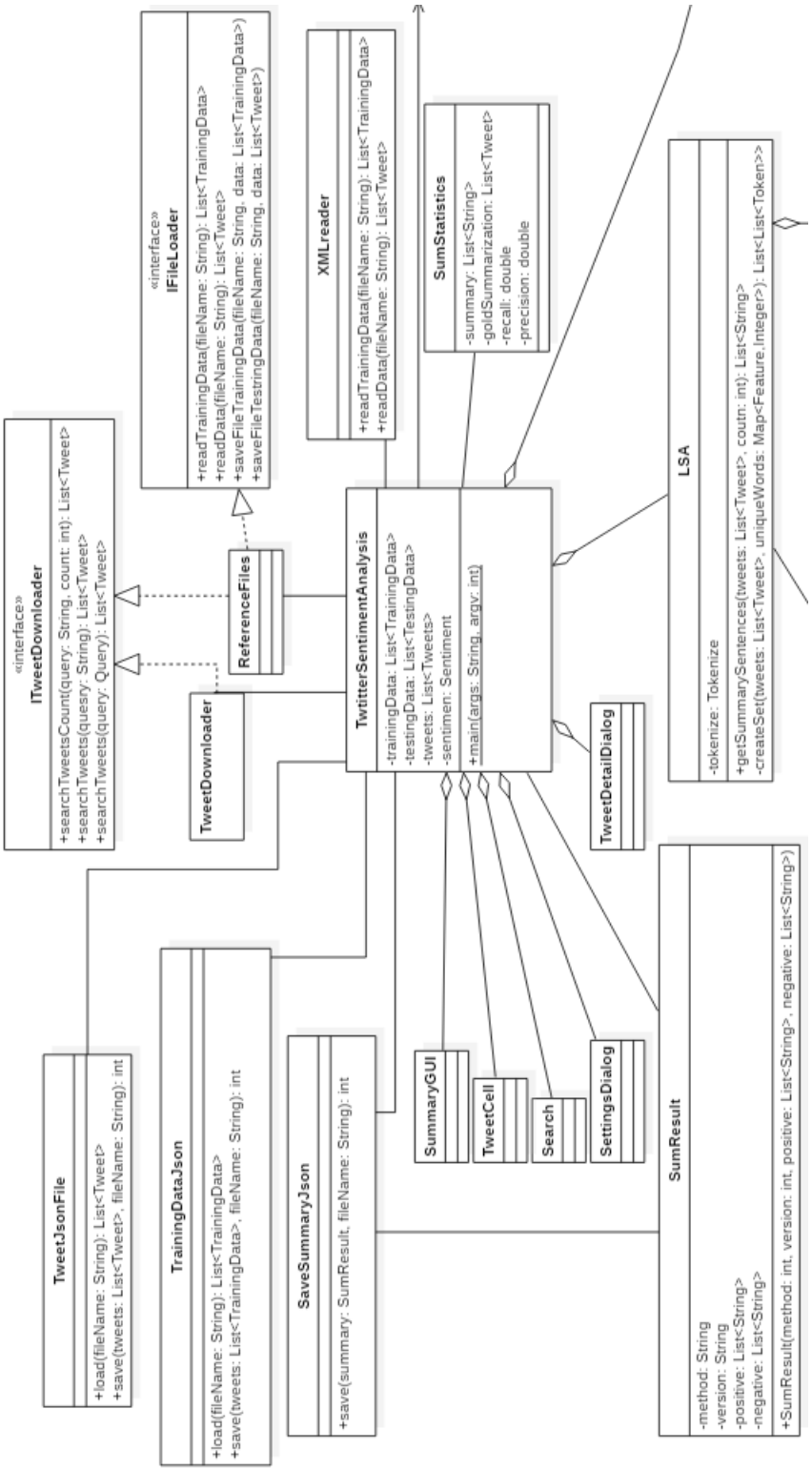
Obrázek 10.5: Výsledky sumarizace tweetů

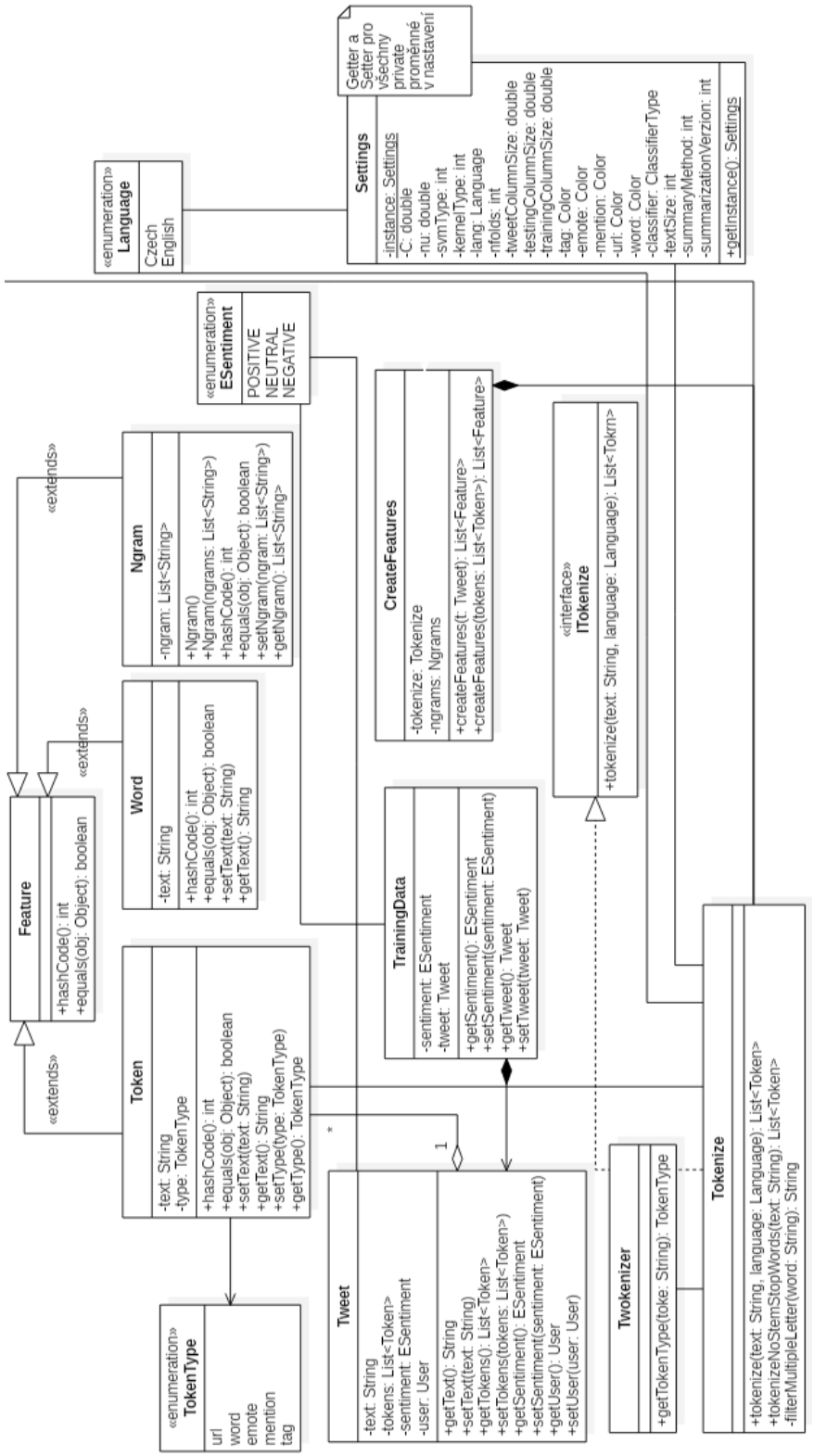
`<nazev>.jar`
`topics`
`summaries-gold`

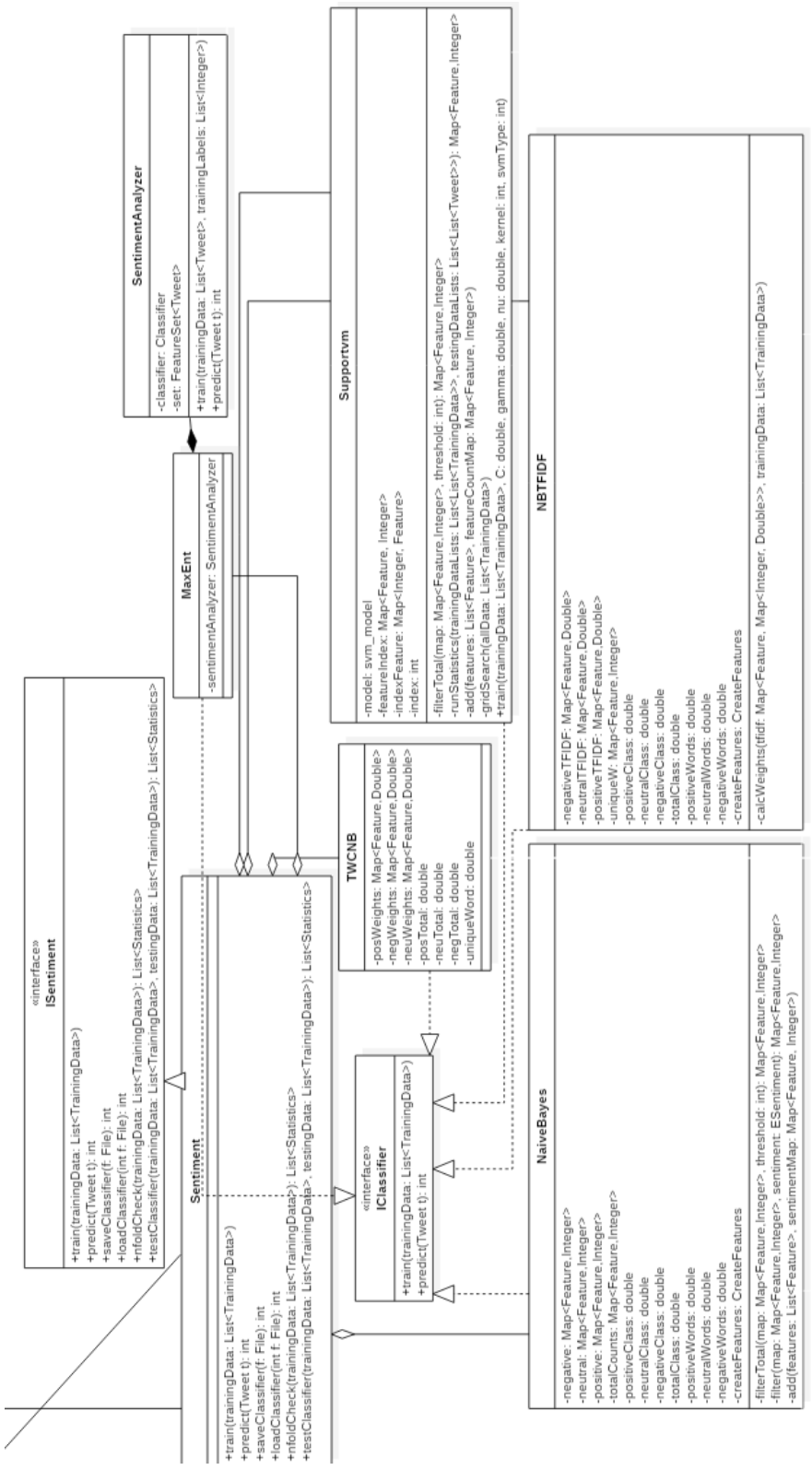
Příloha B: UML diagram tříd

Popis jednotlivých diagramů (kompletní diagram je dostupný na přiloženém disku):

1. Diagram popisující načítání souborů, sumarizaci a GUI
2. Diagram popisující Předzpracování textu a jednotlivé třídy pro ukládání dat
3. Diagram klasifikátorů sentimentu


«interface»
ITweetDownloader
+searchTweetsCount(query: String, count: int): List<Tweet>
+searchTweets(quesry: String): List<Tweet>
+searchTweets(query: Query): List<Tweet>
«interface»
IFileLoader
+readTrainingData(fileName: String): List<TrainingData>
+readData(fileName: String): List<Tweet>
+saveFileTrainingData(fileName: String, data: List<TrainingData>)
+saveFileTestringData(fileName: String, data: List<Tweet>)
XMLreader
+readTrainingData(fileName: String): List<TrainingData>
+readData(fileName: String): List<Tweet>
SumStatistics
-summary: List<String>
-goldSummarization: List<Tweet>
-recall: double
-precision: double
LSA
-tokenize: Tokenize
+getSummarySentences(tweets: List<Tweet>, coutn: int): List<String>
-createSet(tweets: List<Tweet>, uniqueWords: Map<Feature.Integer>): List<List<Token>>
ReferenceFiles
TweetDownloader
TwtitterSentimentAnalysis
-trainingData: List<TrainingData>
-testingData: List<TestingData>
-tweets: List<Tweets>
-sentimen: Sentiment
+main(args: String, argv: int)
TweetDetailDialog
TweetJsonFile
+load(fileName: String): List<Tweet>
+save(tweets: List<Tweet>, fileName: String): int
TrainingDataJson
+load(fileName: String): List<TrainingData>
+save(tweets: List<TrainingData>, fileName: String): int
SaveSummaryJson
+save(summary: SumResult, fileName: String): int
SummaryGUI
TweetCell
Search
SettingsDialog
SumResult
-method: String
-version: String
-positive: List<String>
-negative: List<String>
+SumResult(method: int, version: int, positive: List<String>, negative: List<String>)


Getter a Setter pro všechny private proměnné v nastavení
«enumeration» Language
Czech
English
Settings
-instance: Settings
-C: double
-nu: double
-svmType: int
-kernelType: int
-lang: Language
-nfolds: int
-tweetColumnSize: double
-testingColumnSize: double
-trainingColumnSize: double
-tag: Color
-emote: Color
-mention: Color
-url: Color
-word: Color
-classifier: ClassifierType
-textSize: int
-summaryMethod: int
-summarizationVerzion: int
+getInstance(): Settings
«enumeration» ESentiment
POSITIVE
NEUTRAL
NEGATIVE
Ngram
-ngram: List<String>
+Ngram()
+Ngram(ngrams: List<String>)
+hashCode(): int
+equals(obj: Object): boolean
+setNgram(ngram: List<String>)
+getNgram(): List<String>
«extends»
Word
-text: String
+hashCode(): int
+equals(obj: Object): boolean
+setText(text: String)
+getText(): String
«extends»
Feature
+hashCode(): int
+equals(obj: Object): boolean
«extends»
CreateFeatures
-tokenize: Tokenize
-ngrams: Ngrams
+createFeatures(t: Tweet): List<Feature>
+createFeatures(tokens: List<Token>): List<Feature>
«interface» ITokenize
+tokenize(text: String, language: Language): List<Token>
TrainingData
-sentiment: ESentiment
-tweet: Tweet
+getSentiment(): ESentiment
+setSentiment(sentiment: ESentiment)
+getTweet(): Tweet
+setTweet(tweet: Tweet)
Token
-text: String
-type: TokenType
+hashCode(): int
+equals(obj: Object): boolean
+setText(text: String)
+getText(): String
+setType(type: TokenType)
+getType(): TokenType
«enumeration» TokenType
url
word
emote
mention
tag
Tweet
-text: String
-tokens: List<Token>
-sentiment: ESentiment
-user: User
+getText(): String
+setText(text: String)
+getTokens(): List<Token>
+setTokens(tokens: List<Token>)
+getSentiment(): ESentiment
+setSentiment(sentiment: ESentiment)
+getUser(): User
+setUser(user: User)
Twokenizer
+getTokenType(toke: String): TokenType
Tokenize
+tokenize(text: String, language: Language): List<Token>
+tokenizeNoStemStopWords(text: String): List<Token>
-filterMultipleLetter(word: String): String
*
1


SentimentAnalyzer
-classifier: Classifier
-set: FeatureSet<Tweet>
+train(trainingData: List<Tweet>, trainingLabels: List<Integer>)
+predict(Tweet t): int
MaxEnt
-sentimentAnalyzer: SentimentAnalyzer
Supportvm
-model: svm_model
-featureIndex: Map<Feature, Integer>
-indexFeature: Map<Integer, Feature>
-index: int
-filterTotal(map: Map<Feature,Integer>, threshold: int): Map<Feature,Integer>
-runStatistics(trainingDataLists: List<List<TrainingData>>, testingDataLists: List<List<Tweet>>): Map<Feature,Integer>
-add(features: List<Feature>, featureCountMap: Map<Feature, Integer>)
-gridSearch(allData: List<TrainingData>)
+train(trainingData: List<TrainingData>, C: double, gamma: double, nu: double, kernel: int, svmType: int)
«interface»
ISentiment
+train(trainingData: List<TrainingData>)
+predict(Tweet t): int
+saveClassifier(f: File): int
+loadClassifier(int f: File): int
+nfoldCheck(trainingData: List<TrainingData>): List<Statistics>
+testClassifier(trainingData: List<TrainingData>, testingData: List<TrainingData>): List<Statistics>
Sentiment
+train(trainingData: List<TrainingData>)
+predict(Tweet t): int
+saveClassifier(f: File): int
+loadClassifier(int f: File): int
+nfoldCheck(trainingData: List<TrainingData>): List<Statistics>
+testClassifier(trainingData: List<TrainingData>, testingData: List<TrainingData>): List<Statistics>
TWCNB
-posWeights: Map<Feature,Double>
-negWeights: Map<Feature,Double>
-neuWeights: Map<Feature,Double>
-posTotal: double
-neuTotal: double
-negTotal: double
-uniqueWord: double
«interface»
IClassifier
+train(trainingData: List<TrainingData>)
+predict(Tweet t): int
NBTFIDF
-negativeTFIDF: Map<Feature,Double>
-neutralTFIDF: Map<Feature,Double>
-positiveTFIDF: Map<Feature,Double>
-uniqueW: Map<Feature,Integer>
-positiveClass: double
-neutralClass: double
-negativeClass: double
-totalClass: double
-positiveWords: double
-neutralWords: double
-negativeWords: double
-createFeatures: CreateFeatures
-calcWeights(tfidf: Map<Feature, Map<Integer, Double>>, trainingData: List<TrainingData>)
NaiveBayes
-negative: Map<Feature,Integer>
-neutral: Map<Feature,Integer>
-positive: Map<Feature,Integer>
-totalCounts: Map<Feature,Integer>
-positiveClass: double
-neutralClass: double
-negativeClass: double
-totalClass: double
-positiveWords: double
-neutralWords: double
-negativeWords: double
-createFeatures: CreateFeatures
-filterTotal(map: Map<Feature,Integer>, threshold: int): Map<Feature,Integer>
-filter(map: Map<Feature,Integer>, sentiment: ESentiment): Map<Feature,Integer>
-add(features: List<Feature>, sentimentMap: Map<Feature, Integer>)

Příloha C: Ukázka použití na českém Twitteru

Pro ukázku bylo zvoleno téma Světového poháru v Rusku 2018. Byl proveden dotaz s parametry:

- Search - #worldcup
- Enable location - Ano
- Longitude, Latitude - 15.1, 49.8
- Range - 100
- Language - Čeština
- Count - 500

Bylo nalezeno 230 tweetů. Výsledná sumarizace byla následující. **Negativní:**

#Egypt, který upíral ke svému návratu na #WorldCup tolik nadějí, upadl po druhé porážce do stavu kolektivní deprese. Dokonale to vystihla @monaeltahawy: "Těch 90 minut odráží vše, co je s naší zemí špatně: nedostatek důvěry v sebe sama, mrhání talentem mladých lidí..."#EGY #RUS https://t.co/xZVjbNCS5e #Messi předvedl proti #Iceland neskutečně nervózní výkon a tak #Argentina jen remizovala. Kvůli zbrklosti zahodil penaltu a dva přímáky. Škoda! Nemám #CR7 rád, ale v disciplíně soustředění má

navrch. Holt nad sebou jako Portugalec nemá žádného boha typu #Maradona. #WorldCup https://t.co/zowIJnYr8j A pak už jsou jen Rusáci, které rád vidím na kolenou. #Mexico předvedlo každopádně výborný výkon, zatímco #Germany byli naprosto jaloví. Navíc, jak psal @jindrichsidlo: pro nás fanoušky @Arsenal je osvobozující vidět @MesutOzil1088 flákat se i v národním týmu. #Worldcup https://t.co/PsHdtEHc6z Zdravotní karta Neymara po dnešním zápase: - zlomenina lýtkové kosti - fraktura lebky - natažený stehenní sval - vymknutý kotník - naražená kostrč

Kdyby raději hrál týmově. #BRASUI #WorldCup #msfotbal Politicky se dva největší argentinské deníky @LANACION a @clarincom málokdy shodnou, ale v pohledu na blamáž, kterou předvedla fotbalová reprezentace #ARG v zápase proti #CRO na #WorldCup si vcelku notují: „ostuda, katastrofa, zklamání, jednou nohou venku z šampionátu“ #ARGCRO https://t.co/m4S2kdLFoU

Pozitivní:

#FIFA neodvedla optimální práci. Víme, co předcházelo rozhodnutí pořádat #WorldCup v Rusku. Tehdejší šéf FIFA Sepp Blatter je dnes na šampionátu osobním hostem Vladimira #Putin-a. Doufám, že z toho vyvodíme ponaučení do budoucna: @Telicka v #InterviewPlus @CRoPlus #MStipovačka #WorldCup Vítěz: #FRA Poražený finalista: #ESP Nej. hráč: Mbappe (#FRA) Nej. střelec: Mbappe (#FRA) Nej. gólman: de Gea (#ESP) Překvapení: #ENG Propadák: #ARG Vítěz: Brazílie Poražený finalista: Německo

Nejlepší hráč: Neymar Nejlepší střelec: Neymar Nejlepší brankář: Alisson

Zklamání: Chorvatsko Překvapení: Island

#MStipovačka Ženy v Íránu na fotbal nesmí. Proto jich do Ruska přijela spousta a mistrovství si neskutečně užívají. Je na nich vidět opravdu čistá radost. Reportáž na @Radiozurnal1 ve čtvrtek ve 13:40. #FifaWorldCup2018 #IRN #Iran #women #zeny #mistrovstvisveta #football #WorldCup https://t.co/JZbx4IZx50

Vzhledem k tomu, že trénování proběhlo na tématu, které s fotbalem nesouvisí, jsou dosažené výsledky dle mého názoru dobré. Použitý MaxEnt klasifikátor byl natrénován na datasetu obsahující příspěvky z českého Facebooku, které se týkají hodnocení produktů a služeb.

Příloha D: Soubor s nastavením

```
[classifier]
type = 0

[preference]
font_size = 20
lang = cz

[summary]
method = 0
version = 1

[svm]
c = 10.0
nu = 0.7
kernel = 0
type = 0

[testing]
nfolds = 10

[ui]
tweet_column_width = 400.0
```

```
[colors]
tag = 0x008000ff
mention = 0xa52a2aff
word = 0x000000ff
url = 0x0000ffff
emote = 0xff00ffff
```

Příloha E: Obsah přiloženého disku

- Datasety- obsahuje použité datasety
  - dai twitter
    - data pro testování anglického twitteru
    - formát: sentiment text (odděleno tabulátorem)
    - en_sentiment.csv - kompletní dataset, 3 hodnocení pro každý záznam
    - en_sentiment_modified.csv - dataset, kde se shodují všechny hodnocení
  - facebook cz - data z českého fb, obsahují hodnocení produktů
    - gold-labels.txt - obsahuje snetiment jednotlivých příspěvků
    - gold-posts.txt - texty příspěvků
  - opiniosis - slouží k otestování sumarizace textů. Nutno překopírovat do složky s jar souborem.
    - topics - obsahuje data k sumarizaci, každý soubor = téma
    - summaries-gold - každá složka existuje několik sumarizací pro určité téma
- Latex - obsahuje soubory pro vytvoření textu práce v LaTEX
- Program - obsahuje program

  - Javadoc vygenerovaný javadoc
  - Licence licence k pouitým knihovnám
  - jar obsahuje spustitelný JAR soubor, pokud není dostupný, lze vytvořit spuštěním createwindowsjar.bat (na Windows OS)
  - twittersentimentanalysis - zdrojové kódy programu
  - createwindowsjar.bat - vytvoří spustitelný jar soubor na os windows

- Hain_Jakub_A16N0030P_Poster_2019.pdf - poster v pdf
- Hain_Jakub_A16N0030P_Poster_2019.pub - poster ve formátu pub
- Hain_Jakub_A16N0030P_Diplomová_práce_2019.pdf - kompletní text práce
- uml.mdj - upravitelný v uml diagram v StarUML
- uml.svg - uml diagram tříd ve formátu svg